Development and application of computer-aided design methods for cell factory optimization

Cardoso, Joao

Publication date:
2017

Document Version
Publisher's PDF, also known as Version of record

Development and application of computer-aided design methods for cell factory optimization

A DISSERTATION PRESENTED
BY
JOÃO GONÇALO ROCHA CARDOSO
TO
THE NOVO NORDISK FOUNDATION CENTER FOR BIOSUSTAINABILITY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
IN THE SUBJECT OF
BIOENGINEERING AND SYNTHETIC BIOLOGY

UNDER THE SUPERVISION OF
MIKAEL RØRDAM ANDERSEN
MARKUS J. HERRGÅRD
NIKOLAUS SONNENSCHEIN

TECHNICAL UNIVERSITY OF DENMARK
KONGENS LYNGBY

2017

Acknowledgments

I WOULD LIKE TO THANK my supervisors Nikolaus Sonnenschein, Markus Herrgård and Mikael Rørdram Andersen for their support and guidance during the PhD. As part of the PhD requirements, I did an external stay at Chr. Hansen. I want to acknowledge them for hosting me, specially Ahmad Zeidan and Rute Neves for making this collaboration possible. I had a great time ath Chr. Hansen and learned a lot during that period. I also want to acknowledge Tadas Jakočiūnas and Michael K. Jensen for the opportunity to collaborate in the development of a yeast strain with potential industrial value. Thank you for trusting my computational modeling ideas. Also, I need to thank Kristian Jensen for helping me with the mathematics of things through all the projects I have been involved.

I would like to thank my fellow PhD colleagues Anne Sofie Lærke Hansen and Kristian Jensen for translating my abstract to danish. And, together with Christian Lieven, for proof-reading parts of my thesis. Also, a big thanks to Miguel Campodonico and Sumesh Sukumara for helping finding relevant reports to for the synopsis of this thesis and for proof-reading it too.

I need to thank my parents too, because it wouldn't be possible to be where I am today without their support. They saw me growing up, provided me all I needed, including a good education. They provided financial support that helped me move to Copenhagen where accommodation is very expensive. I need to thank my friends Alfredo "Gigio" Pereira, José Mendanha, David Alvim, Rui "Tika" Moreira, Rodrigo "Piri" Andrade and Vitor "Vitéló" Martins for being there everyday on the other side of the internet with their funny, sometimes charming and most of the times totally weird messages. Also, a special thanks to my best friend, Artur Rebelo for being always there, for the checkup call now and then and for showing up here many times. A big thanks to Pedro Coelho and Mário Pena. They also know what it takes to move abroad and they traveled here too, to see my new life and share experiences in Denmark. I also have to thank my friend Maria João Pena for always being supportive and caring.

During these past three years, I have also made new friends. Steven Jøris, who became my

first friend here in Denmark and with whom I shared many beers, concerts and all sorts of nice events. He also fixed my LaTeX tables in this thesis. Thanks for all of that. Another thanks my buddies Gita Kristiansen, Peter Marshall, Stephanie Emory and Urvish Trivedi who are always up to meet and have some fun! Life in Denmark would have been much boring without them.

I need to thank my Portuguese friends here in Denmark (and some not so Portuguese, but also part of crowd), Tiago Cortez Pinto, Sara Peres Dias, Pedro Correia Monteiro, Patricia Carvalho, Pedro 'Google' Correia, Sofia Loução, João Barroso, Miina Malkki, João Feijó, Sara Jayne, Gustava Seleiro and Angela Carvalho, Sebastian Theobald, Henrique Machado and Klara Bojanovic Machado. Thank you for being good friends and all the moments we spent together. And special thanks to Mafalda Cavaleiro, who has been listening to all of my problems, complaints, and all kinds of non-sense these years. And thanks for helped with proofreading the thesis. To my fellow PhD colleagues (some who already graduated) Maja Rennig, Mathew Fabre, Marta Matos, Isotta D'Arrigo, Patricia Calero Valdayo, Mikkel Lindegaard and Carlotta Ronda. We shared many scientific and less-scientific conversations, it has been a pleasure.

A big thank you to my colleagues in the Sequencing, Informatics, Modeling and Software (SIMS) group for all the good moments shared at CfB. Another special thanks to Anna Koza, for being super welcoming and nice to me since the beginning and a good friend who I can talk to. Also to Emre Özdemir for all the fruitful scientific discussions about modeling. Thanks to my fellow brewers at the Beer Club and footballers at the Football Club for super entertaining evenings of brewing and drinking beer, playing football and drinking beer, or just drinking beer. We drank many beers together. One more thank to Patricia Calero Valdayo for taking care of my cat when I needed to be away. She was a lifesaver.

Finally, I want to acknowledge the founding agencies that supported my PhD. First, the Novo Nordisk Foundation for providing the financial support for my PhD. Also, the Otto Mønsted Fund for supporting the costs of my trip to Japan where I attended a very important conference in metabolic engineering.

English Abstract

Genetically modified organisms (GMOs) can be used to produce chemicals for everyday applications. Engineering microorganisms is a multidisciplinary task comprising four steps: design, build, test and learn. The design and learn phases rely on computational, statistical models, data analysis and machine learning. The process of creating strains with commercially relevant titers is time consuming and expensive. Computer-aided design (CAD) software can help scientists build better strains by providing models and algorithms that can be used to generate and test hypotheses before implementing them in the lab.

Metabolic engineering already uses computational tools to design and analyze the metabolic and regulatory mechanisms of microorganisms. Genome-scale metabolic models (GEMs) describe the biochemical reactions in an organism and their relationship with the genome, hence they can be used to design microbial cell factories. In this PhD thesis we present cameo, a CAD software for metabolic engineering that uses GEMs. State-of-the-art and novel algorithms are implemented in cameo. These algorithms have been made accessible using a high-level API to enable any user to start running them without having advanced programming skills. Using cameo, we designed a *Saccharomyces cerevisiae* strain with improved mevalonate production.

In the food industry, recombinant DNA technologies cannot be used because of strict GMO regulations, especially in Europe. This industry relies on classical strain improvement (CSI) and adaptive laboratory evolution (ALE) to create new and better products. Nevertheless, some engineering and design principles can be applied to create strains in this industrial setup. In this work, we present MARSI, a software tool that uses a completely new model-based approach to strain design, focusing on metabolite targets. MARSI designs can be implemented using ALE or CSI.

We used MARSI to enumerate metabolite targets in *Escherichia coli* that could be used to replace experimentally validated gene knockouts.

Genetic variability occurs naturally in cells. However, the effects of those variations are unpredictable and can impact the performance of production strains. Moreover, strains resulting from CSI and ALE experiments contain a lot of mutations that are not trivial to explain. In this thesis, we explored strategies to integrate re-sequencing data using GEMs. Here, we present a workflow to integrate and analyze data from *E. coli* wild-type, mutant and closely related strains. In this study, we evaluated the effect of genetic variability on $k_{cat}s$. These parameters can be used to constrain GEMs and produce more accurate predictions. Therefore, using a combination of bioinformatics, chemoinformatics and machine learning tools, we explored the landscape of $k_{cat}s$ using multiple enzyme sequences and their chemical reactions.

Danish Abstract

Genmodificerede organismer (GMO) kan anvendes til at producere mange alment brugte kemikalier. Modificering af mikroorganismer er en tværdisiplinel proces bestående af fire faser: design, konstruktion, test og analyse. Design- og analysefaserne anvender statistiske modeller, dataanalyse og machine learning. At udvikle stammer til kommercielt relevante processer er tidskrævende og dyrt. Computer-assisteret design (CAD) software hjælper forskere med at konstruere bedre stammer ved hjælp af modeller og algoritmer, der kan generere og teste strategier før de implementeres i laboratoriet.

Inden for metabolic engineering er computerbaserede værktøjer allerede i brug i forbindelse med design og analyse af mikroorganismers metaboliske og regulatoriske mekanismer. Genomskala metaboliske modeller (GEMs) beskriver alle de biokemiske reaktioner, der finder sted i cellen samt deres forhold til genomet, og kan således anvendes til at designe mikrobielle cellefabrikker. I denne PhD-afhandling præsenteres cameo, et CAD-værktøj til metabolic engineering der er baseret på GEMs og som implementerer både eksisterende og nyudviklede algoritmer. Algoritmerne er tilgængelige via et brugervenligt API, så de kan anvendes selv uden videregående programmeringsfærdigheder. Ved hjælp af cameo har vi designet en *Saccharomyces cerevisiae* stamme med forbedret produktion af mevalonate.

I fødevareindustrien kan rekombinante DNA-teknikker ikke anvendes på grund af streng lovgivning inden for GMO området, specielt i Europa. Industrien anvender derfor klassisk strain-engineering (CSI) eller adaptive laboratory evolution (ALE) til at udvikle nye og forbedrede produkter. På trods af dette kan visse engineering- og design-principper også anvendes til at udvikle stammer inden for denne industri. I denne afhandling præsenterer vi softwaren MARSI, som

anvender en helt ny model-baseret design-metode baseret på metabolit-targets. De designs, der findes af MARSI, kan implementeres in vivo enten via CSI eller ALE. Her demonstreres MARSI ved at finde metabolit-targets i *Escherichia coli*, som kan erstatte eksperimentelt validerede gen-deletioner.

Genetisk variation forekommer naturligt i celler, men effekterne af varianter er næsten umulige at forudsige og kan påvirke produktiviteten af cellefabrikker. Derudover kan der i stammer, skabt ved hjælp af CSI eller ALE, være opstået mutationer som ikke uden videre kan forklares. I denne afhandling udforskes strategier til at integrere sekventeringsdata med GEMs. Vi præsenterer et workflow til at analysere data fra *E. coli* vildtype- og mutantstammer samt nært beslægtede stammer. Ydermere evalueres effekten af genetiske variationer på enzymers k_{cat}-parametre. Disse parametre kan bruges til at opstille restriktioner i GEMs for derved at lave bedre forudsigelser af metabolisk aktivitet. Ved hjælp af en kombination af bioinformatik, kemoinformatik, statistik og machine learning, udforskede vi forskellige enzymers k_{cat} ved at inddrage deres sekvenser og kemiske reaktioner.

List of Publications

Publications included in this thesis:

1. João G.R. Cardoso, Ahmad Zeidan, Kristian Jensen, Nikolaus Sonnenschein, Ana R. Neves and Markus J. Herrgård (2017) MARSI: metabolite analogues for rational strain improvement Bioinformatics (submitted)

2. João G. R. Cardoso, Kristian Jensen, Christian Lieven, Anne Sofie Laerke Hansen, Svetlana Galkina, Moritz Emanuel Beber, Emre Ozdemir, Markus J. Herrgard, Henning Redestig, Nikolaus Sonnenschein (2017) *Cameo*: A Python Library for Computer Aided Metabolic Engineering and Optimization of Cell Factories bioRxiv 147199 doi:10.1101/147199

3. João G. R. Cardoso, Mikael R. Andersen, Markus J. Herrgård and Nikolaus Sonnenschein (2015) Analysis of genetic variation and potential applications in genome-scale metabolic modeling Front Bioeng Biotechnol 3: 1–12. doi: 10.3389/fbioe.2015.00013

Other publications:

1. Henrique Machado, João G.R. Cardoso, Sonia Giubergia, Kristoffer Rapacki and Lone Gram (2017) FurIOS: A Web-Based Tool for Identification of *Vibrionaceae* Species Using the fur Gene. Front Microbiol 8: 1–8.

2. Kristian Jensen , João G.R. Cardoso, Nikolaus Sonnenschein (2017) Optlang: An algebraic modeling language for mathematical optimization. J Open Source Softw 2.

3. Isotta D'Arrigo, João G. R. Cardoso, Maja Rennig, Nikolaus Sonnenschein, Markus J. Herrgård, Katherine S. Long (2017) Systems analysis of the Pseudomonas putida growing on non-trivial carbon sources using transcriptomics and genome-scale modeling. Appl Environ Microb (submited)

Author List

Chapter 1: João G. R. Cardoso

Chapter 2: João G. R. Cardoso, Kristian Jensen, Christian Lieven, Anne S. L. Hansen, Svetlana Galkina, Moritz E. Beber, Emre Ozdemir, Markus J. Herrgård, Henning Redestig and Nikolaus Sonnenschein

Chapter 3: João G. R. Cardoso, Ahmad A. Zeidan, Kristian Jensen, Ana Rute Neves, Nikolaus Sonnenschein and Markus J. Herrgård

Chapter 4: Tadas Jakočiūnas[§], João G. R. Cardoso[§], Nikolaus Sonnenschein, Markus J. Herrgård, Jay D. Keasling and Michael K. Jensen

Chapter 5: João G. R. Cardoso, Mikael Rørdam Andersen, Markus J. Herrgård and Nikolaus Sonnenschein

Chapter 6: João G. R. Cardoso, Nikolaus Sonnenschein and Markus J. Herrgård

Chapter 7: João G. R. Cardoso

[§]shared co-authorship

Abbreviations

ALE adaptive laboratory evolution

BiGG Biochemical Genetic and Genomic

CAD computer-aided design

cameo computer-aided metabolic engineering and optimization

CDS coding sequence

CLI command line interface

COBRA Constraint-Based Reconstruction and Analysis

CRISPR Clustered Regularly Interspaced Short Palindromic Repeats

CSI classical strain improvement

CTM Chemical Translation Service

DHA dihydroxyacetone

DMS deep mutational scanning

DoE design of experiment

EC Enzyme Commission

EFM elementary flux mode

EFMA elementary flux mode analysis

FACS fluorescence-activated cell sorting

FBA Flux Balance Analysis

FPP farnesyl diphosphate

FVA Flux Variability Analysis

FVSEOF flux variability scanning based on enforced objective flux

GA genetic algorithm

GC gas chromatography

GEM genome-scale metabolic model

GIM3E gene inactivation moderated by metabolism, metabolomics and expression

GSM genome-scale metabolic model

GMO genetically modified organism

GPR gene-protein-reaction

GRAS generally recognized as safe

GRAVY grand average of hydropathy

GUI graphical user interface

GWAS genome-wide assotiation studies

HMG-CoA 3-hydroxy-3-methylglutaryl-CoA

HTS high-throughput screening

IOMA integrative omics-metabolic analysis

LASSO least absolute shrinkage and selection operator

LASSO-LARS LASSO model fit with least-angle regression

LARS least-angle regression

LC liquid chromatography

LP linear programming

HMM hidden Markov Models

HPC high performance computing

lMOMA Linear Minimization of Metabolic Adjustment

MAGE multiplex automated genome engineering

MARSI metabolite analogues for rational strain improvement

MCS minimal cut sets

MILP mixed-integer linear programming

MiMBl Minimization of Metabolites Balance

MS mass spectrometry

MOMA Minimization of Metabolic Adjustment

NGS Next-Generation Sequencing

NPSA non-polar surface area

PCA principal component analysis

pFBA Parsimonious Enzyme Usage Flux Balance Analysis

PLA polylactic acid

PPI protein-protein interaction

PPP pentose phosphate pathway

PSA polar surface area

PSSM position-specific scoring matrix

PTM post-translational modification

QP quadratic programming

RBS ribosome binding site

rDNA recombinant DNA

ROOM Regulation on/off Minimization

RSS residual sum of sequares

SC synthetic complete

SIFT Sorting Intolerant from Tolerant

SNP single nucleotide polymorphism

SNV single nucleotide variation

SS structural score

TC Tanimoto coefficient

WGS whole genome sequencing

YPD yeast extract peptone dextrose

Synopsis

The evolution of modern societies lead to the introduction of chemicals in goods and services we use everyday. Chemicals can be found all around everything we interact with, including plastics, cosmetics, soaps, detergents, clothes, and drugs, among others. Moreover, there are many other chemicals that reach us indirectly, which are used in agriculture, energy and heat production, transport of goods and manufacture. The chemical industry is large and valuable. In 2015, it recorded 3.5€ Trillion in sales (The European Chemical Industry Council, 2016).

Most chemicals we use today are produced from feedstocks based on fossil resources (such as, petroleum and natural gas) In Europe, approximately 90% of the chemicals are derived from fossil resources(Bazzanella and Ausfelder, 2017, p 23). With the expected growth of the global population and the industrialization of underdeveloped countries, the demand for chemicals will certainly increase. However, fossil based feedstocks are a nonrenewable resource.

There are, however, alternative feedstocks that can be used to produce some of the chemicals we use. Biomass (i.e., residues from agriculture, wood, crops or seaweed) provides an alternative raw material for chemical production Microorganisms can be used to convert the biomass into these chemicals in biorefineries, by converting them into cell factories.

Switching to bio-based production of chemicals reduces the dependency on fossil fuels. Because the chemical industry is a valuable and indispensable business, new supply chains have a chance to become profitable business, as access to petroleum reserves becomes more expensive.

Some of these chemicals can and are already being produced from renewable feedstocks (Biddy et al., 2016).

During the past 40 years, recombinant DNA (rDNA) technologies enabled us to combine genetic material from multiple sources to create microorganisms capable of producing chemicals in a fermentor (Adrio and Demain, 2010). Moreover, DNA sequences and laboratory equipment became cheaper an easy to use, reducing the time and cost of creating cell factories. Still, the creation of a commercial relevant strain is still expensive and time consuming (Figure 1).

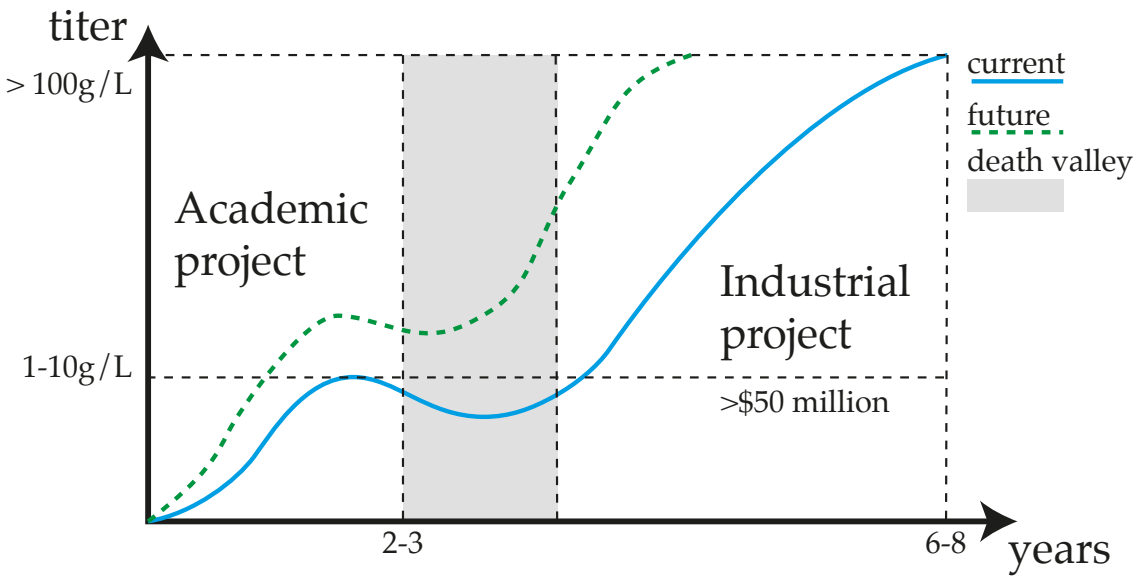


Figure 1: Current titers, time and costs of producing a chemical using biotechnology. Academic projects run in a small scale, mostly for proof of concept. They have low titers and yields. Commercial projects take longer (6-8 years) and require many more people working on it. The cost of bringing a strain to production is over 50$ million (Nielsen and Keasling, 2016). Many projects die in the transition between academia and industry, due to the time and cost of increasing the titers to commercially competitive values and scale-up. In the future, we expect to be faster and better at build strains in the academic environment but also at translating the technology to industry.

The process of creating a cell factory comprises four steps: design, build, test and learn (Nielsen

and Keasling, 2016). To create commercially viable and competitive solutions we need better technology. Reliable computational models and algorithms and cost effective experimental methods and protocols, are constantly being developed to speed up the process and further reduce the costs of building strains.

This PhD thesis address some challenges of designing cell factories using genome-scale metabolic models (GEMs). We propose three major objectives:

1. Create a robust and reliable computer-aided design (CAD) platform for metabolic engineering using GEMs.
2. Make strain design methods available for academic and industrial applications.
3. Use omics data, including resequencing data, to understand the effects of engineering strains and make better designs.

This work is divided in two major parts. The first part comprises the development and application of CAD methods for metabolic engineering, using GEMs. Such methods are capable of enumerate and prioritize metabolic engineering targets. A new method was included, that can be applied in industrial setups where recombinant DNA (rDNA) technologies cannot be used. And, the second part aims to provide a platform for integration and analysis of omics data. We focused our efforts on resequencing data, because little has been done to combine such data with GEM in a systematic fashion, like RNA-Seq or proteomics data.

In the first part of this thesis I focused on CAD methods. To create a CAD tool useful for designing strains, I and my colleagues developed a software package named *computer-aided metabolic engineering and optimization (cameo)*. It contains state-of-the-art and novel algorithms that use GEMs for *in silico* design of cell factories. GEMs are mathematical models that describe the biochemical portfolio of microorganisms. These reactions are connected to the genome and pro-

teome via gene-protein-reaction (GPR) associations (O'Brien et al., 2015). The major tasks implemented in *cameo* are: identification of the most efficient heterologous pathways for a variety chemical products; enumeration of gene knockouts targets, over- and down-regulation targets, and cofactor swap targets; prioritization and analysis of genetic engineering strategies (Figure 2). The software is built in a modular way, the algorithms are optimized for speed and scalability, and it is easy to extend, meaning that new algorithms can be easily implemented. This software aims to be a reference tool for the metabolic engineering community.

Using *cameo*, we designed a *Saccharomyces cerevisiae* strain with improved mevalonate production. The mevalonate pathway is very important, because it leads to the production of a large range of valuable chemicals including fuels, antibiotics, natural colors and anti-cancer drugs, among other products (Liao et al., 2016, Zhang et al., 2011). The strain was built in the laboratory by our colleagues at the Synthetic Biology Tools for Yeast group. This work shows that CAD methods provide useful insights for metabolic engineering and that our software is capable of provide guidance in the design process.

To finalize the first part, we addressed a limitation of the current CAD algorithms: they are meant to be implemented using genetic engineering. However, the European regulation on genetically modified organisms (GMOs) restricts the use of rDNA technologies in the food industry. Food containing GMO organisms, such as yogurt, wine, beer, or fermented meat cannot be commercialized. To stay competitive and create new products, this industry relies on classical strain improvement (CSI) and adaptive laboratory evolution (ALE). Despite the advances in these techniques, they lack the control and speedy provided by rDNA technologies. Nevertheless, developing strains using CSI and ALE can be achieved with using rational design. In collaboration with Chr. Hansen A/S, we developed metabolite analogues for rational strain improve-

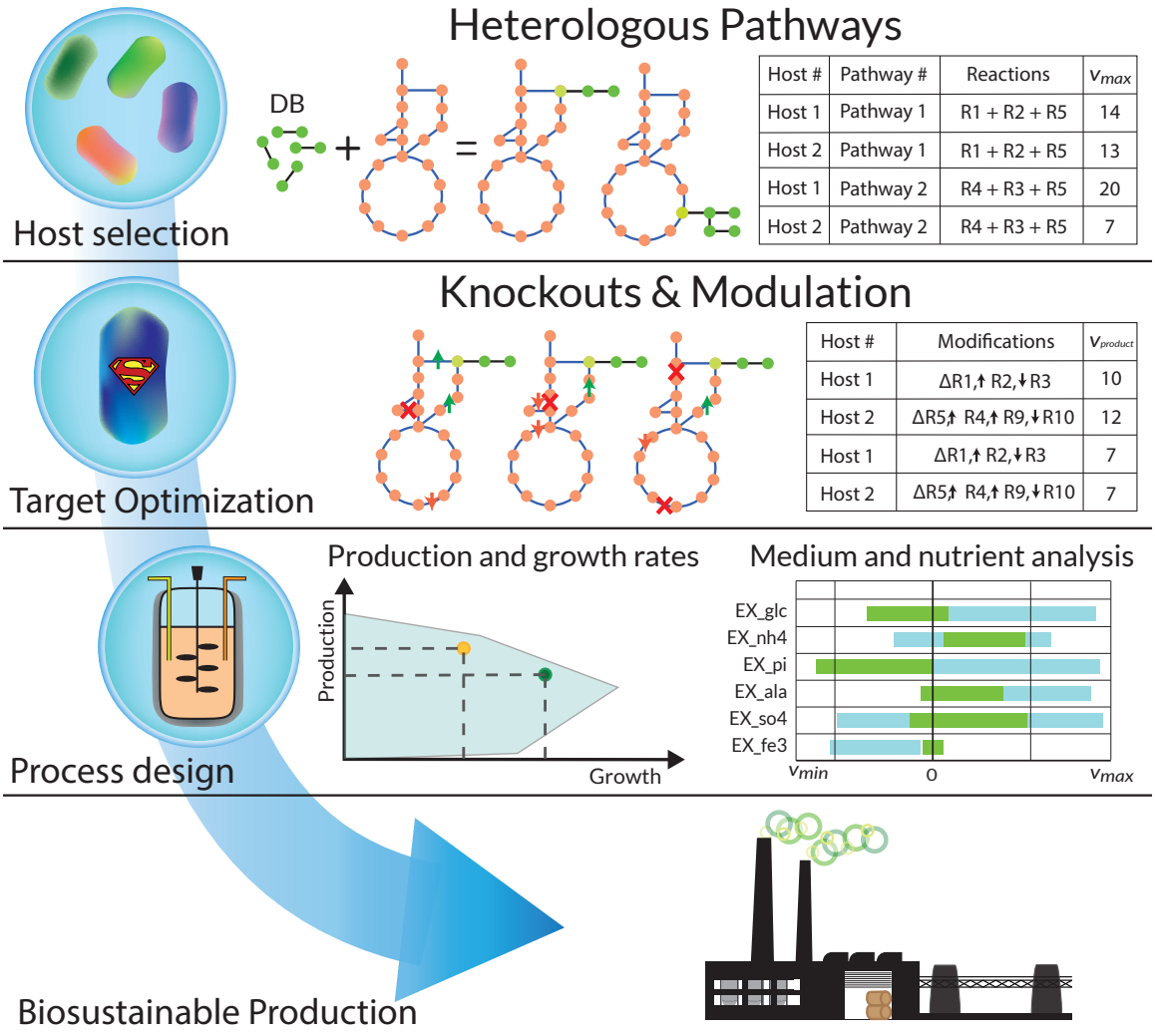


Figure 2: Overview of GEM applications in the context of cell factory optimization. Here, we highlighted three applications of GEMs in the cell factory optimization process. First, the selection of the best performing host can be identified within the limits of stoichiometric boundaries. Second, the identification and prioritization of genetic engineering targets to redirect the flux towards the product. Finally, comparison between growth and production rates, and medium compositions of different strategies to help designing the fermentation process.

ment (MARSI), a new method to identify metabolite targets that can reshape the metabolism of microbial strains towards desired phenotypes. Traditional metabolic engineering software predicts gene or reaction knockout targets. MARSI predicts antimetabolite targets, that can be im-

plemented using ALE or CSI by culturing the cells with chemical analogues. These strains built in the laboratory are compatible with GMO regulations. The software containing this method is an extension of *cameo*.

In the second part of this thesis, we also address the challenges of integrating resequencing data with GEMs. There are several algorithms available to integrate many types of omics, mostly transcriptomics and proteomics, and also metabolomics. In collaboration with my colleagues, we created another software package name *driven* — data-driven constraint-based modeling. Using the same design principles as in *cameo*, this package contains state-of-the-art methods that use high-throughput data to constraint GEMs.

Contrary to most omics data, resequencing data has not yet been combined with GEMs to design cell factories. Strains generated using ALE or CSI contain mutations resulting from evolution. Unlike engineered strains, making sense of these mutations is a challenging task, because their effect (i.e., their contribution to the phenotype) is not always obvious.

I and my colleagues collected tools and knowledge generated from medical applications (i.e., effect of mutations in cancer and other diseases) and adapted those methods to include in our constraint-based modeling framework. First we collected data from SABIO-RK containing detailed kinetic information for many enzymes and reactions, across multiple species. Second, we used SIFT and FoldX to relate the properties of the sequence to the $k_{kact}s$. Finally, we generated features for the reactions and enzymes and applied machine learning algorithms to try to predict the impact of genetic modifications on the $k_{kact}s$ (Figure 3).

We identified no correlation between sequence conservation and catalytic rate. The stability of proteins can, in some cases, inform about the $k_{kact}s$, probably because it relates with the fraction of correctly folded enzyme. Finally, we cannot used our linear model to predict $k_{kact}s$ because the

amount of data that we could retrieve consistently is not sufficient.

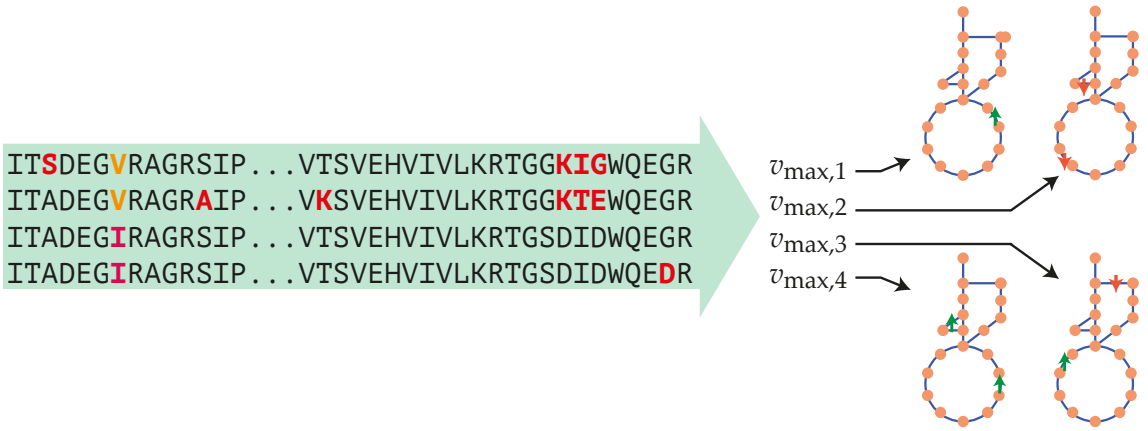


Figure 3: Predicting flux limits imposed by genetic variation. The gene sequence (and consequently the amino-acid composition) of an enzyme will have consequences on the amount of expressed protein, protein stability and activity. If we can predict those parameters, we can then use the v_{max} of the reactions in a GEM and better understand the effect of mutations in enzymes.

References

Jose-Luis Adrio and Arnold L. Demain. Recombinant organisms for production of industrial products. *Bioengineered Bugs*, 1(2):116–131, mar 2010. ISSN 1949-1018. doi: 10.4161/bbug.1.2.10484.

Alexis Michael Bazzanella and Florian Ausfelder. Low carbon energy and feedstock for the european chemical industry. Technical report, DECHEMA Gesellschaft für Chemische Technik und Biotechnologie e.V., Frankfurt am Main, Germany, 2017. URL https://dechema.de/dechema_media/Technology_study_Low_carbon_energy_and_feedstock_for_the_European_chemical_industry-p-20002750.pdf.

Mary J. Biddy, Christopher Scarlata, and Christopher Kinchin. Chemicals from biomass: A market assessment of bioproducts with near-term potential. Technical Report NREL/TP-5100-65509, National Renewable Energy Laboratory, Golden, Colorado, 2016. URL https://www.nrel.gov/docs/fy16osti/65509.pdf.

Pan Liao, Andréa Hemmerlin, Thomas J. Bach, and Mee Len Chye. The potential of the mevalonate pathway for enhanced isoprenoid production. *Biotechnology Advances*, 34(5):697–713, 2016. ISSN 07349750. doi: 10.1016/j.biotechadv.2016.03.005.

Jens Nielsen and Jay D Keasling. Engineering Cellular Metabolism. *Cell*, 164(6):1185–1197, 2016. ISSN 0092-8674. doi: 10.1016/j.cell.2016.02.004.

Edward J. O'Brien, Jonathan M. Monk, and Bernhard O. Palsson. Using genome-scale models to predict biological capabilities. *Cell*, 161(5):971–987, 2015. ISSN 10974172. doi: 10.1016/j.cell.2015.05.019.

The European Chemical Industry Council. The european chemical industry facts & figures 2015. Technical report, The European Chemical Industry Council, Brussels, Belgium, 2016. URL http://fr.zone-secure.net/13451/186036/#page=1.

Fuzhong Zhang, Sarah Rodriguez, and Jay D. Keasling. Metabolic engineering of microbial pathways for advanced biofuels production, 2011. ISSN 09581669.

Thesis Structure

CHAPTER 1: DESIGNING CELL FACTORIES USING GENOME-SCALE MODELS

The first chapter of this thesis is a brief introduction to cell factory optimization and CAD. It covers their applications and the process of building cell factories. It also introduces constraint-based modeling and how it can be used in the process of designing and analyzing cell factories.

CHAPTER 2: CAMEO: A PYTHON LIBRARY FOR COMPUTER AIDED METABOLIC ENGINEERING AND OPTIMIZATION OF CELL FACTORIES

The second chapter describes the usage and implementation of a CAD software to design cell factories. This library harnesses the power of GEMs and combines them with a collection of state-of-the-art and novel optimization algorithms to enumerate and prioritize genetic intervention strategies in the context of cell factory design.

CHAPTER 3: MARSI: METABOLITE ANALOGUES FOR RATIONAL STRAIN IMPROVEMENT.

Chapter three describes new algorithms for CAD in a context where genetic engineering is not allowed due to GMO regulations. These algorithms predict metabolites targets that can be used to improve strain using CSI or ALE. The software also provides sensitivity analysis to assess im-

pact of metabolite analogues in the metabolism. The software includes a database of chemical analogues and a simple query method to identify metabolite analogues for the metabolite targets.

CHAPTER 4: IMPROVING MEVALONATE PRODUCTION IN *SACCHAROMYCES CEREVISIAE* USING CONSTRAINT BASED MODELING

Chapter four is a research paper describing how to design and build a strain with improved mevalonate production. It describes how CAD algorithms can be used to identify bottlenecks in metabolism and how to test different hypothesis using the GEMs.

CHAPTER 5: ANALYSIS OF GENETIC VARIATION AND POTENTIAL APPLICATIONS IN GENOME-SCALE METABOLIC MODELING

Chapter five is a review about genetic variation, its impact and applications in metabolic engineering. This chapter contains an overview of the existing methods to analyze genetic variants and how it can be used in combination with genome-scale metabolic models for cell factory optimization.

CHAPTER 6: PREDICTING ENZYME KINETICS: THE QUEST FOR THE HOLY GRAIL

The work described on chapter six is an analysis of the kinetics data landscape. It highlights the challenges of collecting and processing the available data in the public domain. Predicting the effects of mutations in the kinetics could provide a better understanding of genetic variability across strains and species, but unfortunately the amount of available data is scarce.

CHAPTER 7: CONCLUSIONS AND PERSPECTIVES

The last chapter summarizes the results of this PhD thesis and the impact of this work in biotechnology. It also contains future perspectives regarding the work presented.

Contents

List of figures

List of tables

In nature nothing is created, nothing is lost, everything changes.

Antoine Lavoisier

1

Designing cell factories using genome-scale models

Summary

This chapter is an introduction to metabolic engineering of cell factories. It is written in three parts. The first describes biotechnology and provides historical context for metabolic engineering. The second part highlights the principles and challenges of metabolic engineering. The third part describes constraint-based modeling and how it can be used for computer-aided design of cell factories.

Introduction

Biotechnology has been defined as the use of biological systems to modify and improve processes and products (United Nations, 2001). Mankind has been using biotechnology for over ten millennia (Figure 1.1). It allowed our species to thrive and modern life would be unimaginable without it. Even the most ancient biotechnology applications created by mankind, like selective breeding and fermentation, are still in practice today.

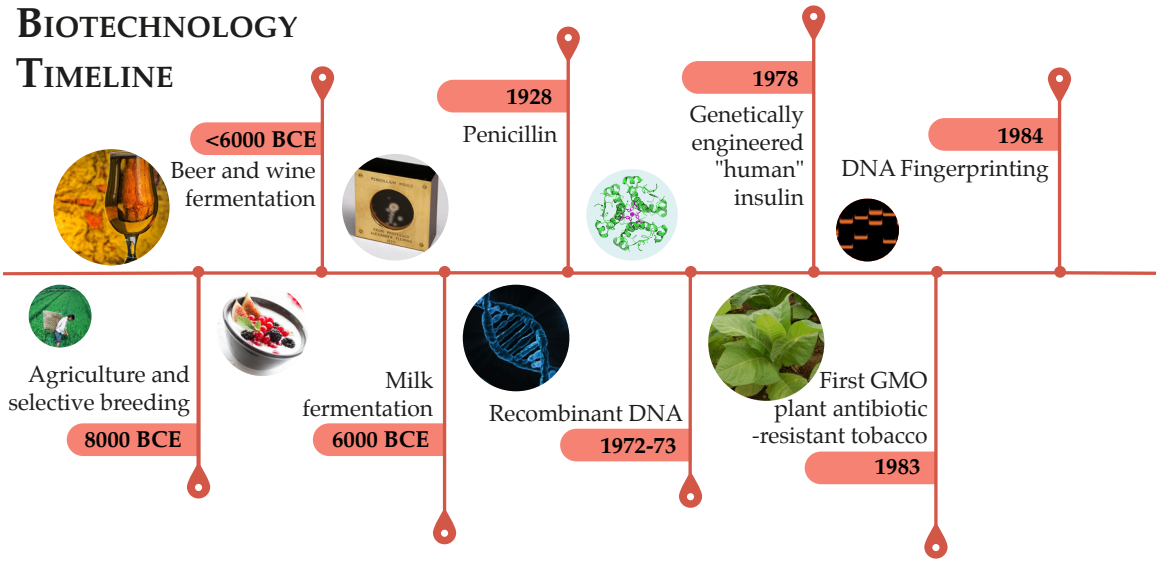


Figure 1.1: Timeline of biotechnological applications. Some of the most relevant biotechnology achievement in time. Agriculture and selective breeding are the most ancient applications of biotechnology, followed by beer (Young, 2017) wine (Hames, 2012, p17), and yogurt (Tribby, 2009). The rDNA technology (Cohen et al., 1973, Jackson et al., 1972, Lobban and Kaiser, 1973, Mertz and Davis, 1972) brought a new dimension to biotechnology by enabling direct manipulation of DNA to create new biotechnological applications.

The first biotechnological products appeared by chance, when microorganisms in nature colonized grapes, ground cereals or milk exposed to the environment and converted the sugars into other chemicals. Inside their cells, microbial species contain enzymes that can catalyze chemical reactions responsible for converting the sugars into ethanol or lactic acid, leading to the production of fermented products such as beer, wine and yogurt. These chemical reactions — required to sustain life within cells — are called metabolism. In fact, cells can convert not only sugars but a variety of other chemicals that may be present in their environment.

Rearranging the metabolism enables us to use microorganisms for chemical production. The first attempts to improve the production of chemicals using microorganism were performed using media optimization and classical strain improvement (CSI) (Rowlands, 1984), e.g., the production of penicillin in around 1940-50's (Barreiro et al., 2012). However, creating strains using CSI is not a very reproducible process and it requires testing a lot of strains without knowing or controlling the changes in their DNA. Nevertheless, CSI is still used nowadays, e.g., in the food industry where legislation in some parts of the world prohibits products containing GMO on the market (Derkx et al., 2014).

Rational improvement of microorganisms became possible in the early 70's with the appearance of rDNA — DNA molecules created in the laboratory with genetic material from multiple sources (Cohen et al., 1973, Jackson et al., 1972, Lobban and Kaiser, 1973, Mertz and Davis, 1972). The rDNA brought a new dawn to biotechnology, by expanding the portfolio of biotechnological applications.

Metabolic engineering

In the beginning of the 90's, the field of metabolic engineering emerged. The term metabolic engineering was coined by Bailey et al. (1990) to describe the application rDNA to improve the production of metabolites and proteins, via manipulation of metabolic and regulatory processes inside the cells.

Metabolic engineering allowed us to create microorganisms that are more efficient at producing chemicals. Some chemicals are part of the native metabolism (Rossum et al., 2016a, Zhang et al., 2009) or produced by other microorganisms. Others are traditionally harvested from plants, such as natural colors, flavors and drugs (Lau and Sattely, 2015, Lee and Schmidt-Dannert, 2002, Marienhagen and Bott, 2013) or animals, such as insulin (Quianzon and Cheikh, 2012). Nowadays, the portfolio of products that can be made using cell factories ranges from cheap bulk chemicals like plastics and fuels to high-value products like drugs and food supplements (Figure 1.2).

Bio-based production of chemicals brings several advantages and these processes tend to be more sustainable. Applications of cell factories can tackle different problems, e.g., CO_2 fixation (Nevin et al., 2011) to decrease green house effect, biofuel production to ease the dependence on fossil fuels (Peralta-Yahya et al., 2012) or biosynthesis of plant derived chemicals, such as opioids,

to reduce the land usage (Galanie et al., 2015). Cell factories can also be used to produce new pharmaceuticals and overcome antibiotic resistance (Weber et al., 2015). Finally, bio-based production can introduce new products to the market with better properties (Yang et al., 2016) to improve our lifestyle.

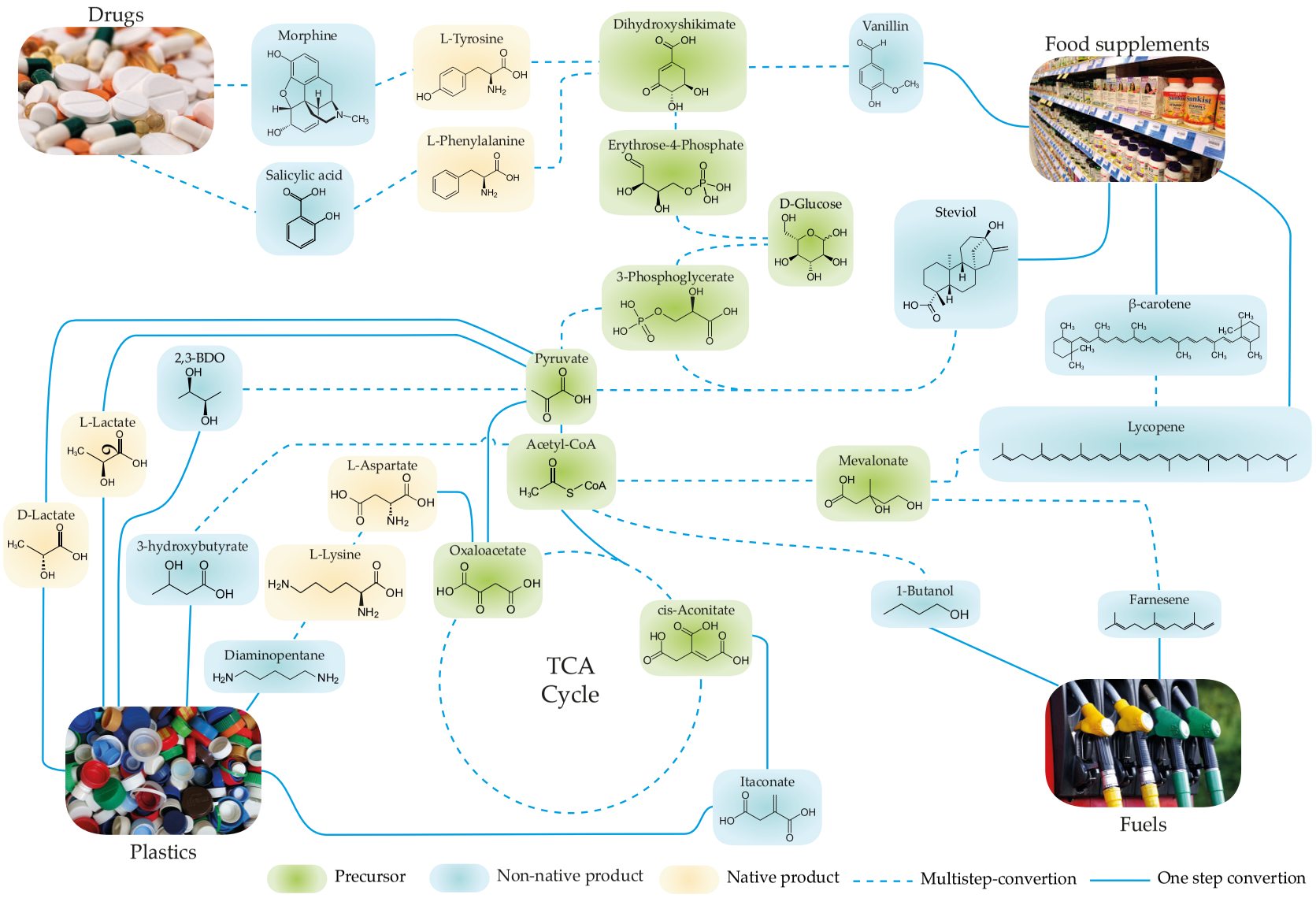
Manipulating microorganisms with commercially relevant yields has proven to be a challenging task. And expensive, too. There are unanticipated effects of applying genetic modifications to microorganisms: deregulation of cell cycle, decreased growth rate, change in protein concentration or decrease of production rates. Also, when cloning heterologous genes in a new environment, it is not guaranteed that the proteins will fold correctly, find their expected substrates or unexpectedly react with native molecules (Bailey, 1991). All of these unpredictable effects can make the development of cell factories a long and expensive process. Currently, creating a commercial production strain requires 6–8 years and over $50 million (Nielsen and Keasling, 2016).

To overcome the uncertainties of engineering cells and enable better implementation of cell factories, Bailey (1991) proposed an iterative cycle consisting of three phases: design, build and test. With the appearance of better and cheaper high-throughput technologies, both laboratorial and computational, a learning step was introduced, where the data generated at each iteration can be used to improve the design process (Figure 1.3).

Design Phase

The aim of the design phase is to identify strategies that can improve the production of a given chemical. It starts with the identification of candidate hosts based on previous knowledge about the process (e.g., temperature, pH and product extraction), raw-materials (e.g. types of sugars or feedstocks), strain physiology (growth-rate, tolerance to product and feed stock toxicity) and genetic engineering tools available. Once the host has been chosen, it is necessary to engineer

Figure 1.2 *(following page)*: Examples of chemical compounds that can be produced using cell factories. Heterologous steps are defined considering classical hosts for metabolic engineering: *Escherichia coli* and *Saccharomyces cerevisiae*. Target compounds are: diaminopentane (bio-nylon) (Kind et al., 2014); Vanillin (flavor) (Hansen et al., 2009); Steviol (sweetener) (Brandle and Telmer, 2007); β-Carotene (natural color) (Li et al., 2013); Lycopene (natural color) (Farmer and Liao, 2001); L-Lactate (plastic, polylactic acid (PLA)) (Dien et al., 2001); D-Lactate (plastic, PLA) (Zhou et al., 2003); 2,3-Butanodiol (fuel) (Ng et al., 2012); 1-Butanol (fuel) (Gulevich et al., 2012); Farnesene (Jet fuel) (Rupasinghe et al., 2001); 3-Hydroxybutyrate (polyester) (Mahishi et al., 2003)


Drugs
Morphine
L-Tyrosine
Dihydroxyshikimate
Vanillin
Food supplements
L-Phenylalanine
Salicylic acid
Erythrose-4-Phosphate
D-Glucose
Steviol
3-Phosphoglycerate
β-carotene
2,3-BDO
Pyruvate
L-Lactate
Acetyl-CoA
Lycopene
L-Aspartate
Mevalonate
D-Lactate
3-hydroxybutyrate
L-Lysine
Oxaloacetate
1-Butanol
Farnesene
cis-Aconitate
Diaminopentane
TCA
Cycle
Itaconate
Plastics
Fuels
Precursor
Non-native product
Native product
Multistep-convertion
One step convertion

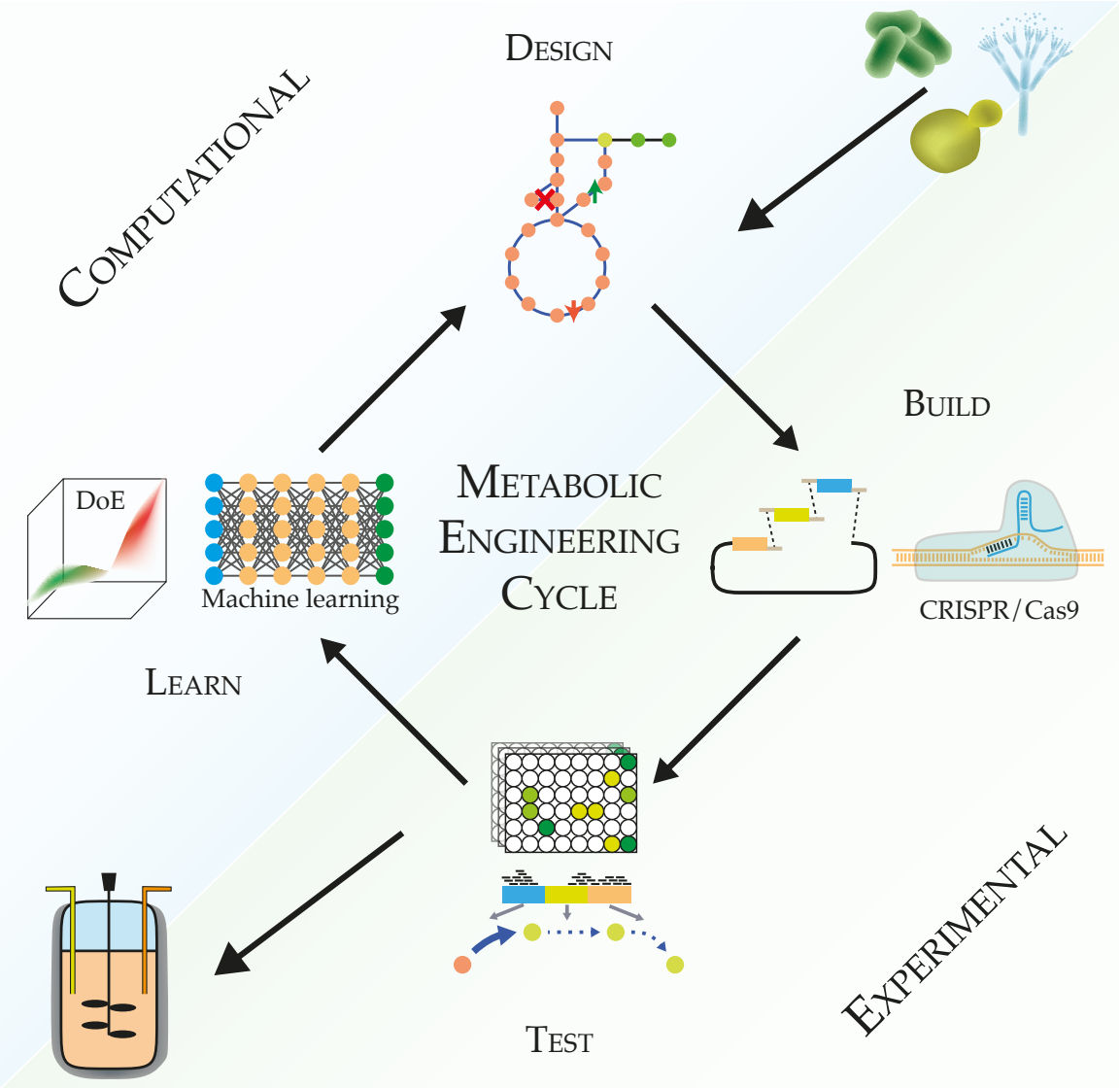


Figure 1.3: The modern metabolic engineering cycle. It consists of four phases: design, build, test, and learn. 1) *Design phase.* Mathematical models, previous knowledge and rational design are combined to make the best strains. 2) *Build phase.* The design is implemented into a host. This includes assembly of heterologous pathways, manipulation of native genes and promoters and/or evolution of strains under selective pressure. 3) *Test phase.* The performance of the implemented strains is evaluated. This is the phase where production is measured. 4) *Learn phase.* Before moving to the next round of design and trying to further improve the strains, the data generated can be used understand what failed and what worked in the current designs. The first and fourth steps rely on computational methods, whereas steps two and three are heavily experimental. This iterative loop stops when the desired titers and yields have been reached.

the metabolism. When the native metabolism is not enough to create the desired product, heterologous elements can be added to connect the native metabolism with the final product or to overcome native regulatory limitations, such as feedback inhibition mechanisms (Nielsen and Keasling, 2016).

Identification of heterologous proteins can be achieved by collecting data from literature or databases. In some cases, where there is not enough information to bridge the native metabolism of the host to a desired product, retro-biosynthesis algorithms can be used to identify *de novo* production pathways (Campodonico et al., 2014, Carbonell et al., 2011, Hatzimanikatis et al., 2005). The biosynthesis of some secondary metabolites, such as tetracycline (an antibiotic) or epothilone B (an anticancer drug) requires complex gene clusters, which involve large sets of genes. Those genes can be found using computation tools capable of identifying these biosynthetic gene clusters from chemical structures (Dejong et al., 2016).

Models of metabolism and regulation are very useful in the design process. Cells are intrinsically complex systems where many species can participate in multiple processes, which makes the selection of genetic manipulations a challenging task. Stoichiometric models, such as genome-scale metabolic models (GEMs) describe the portfolio of biochemical reactions present in an organism and their connection to genes and enzymes, using Boolean rules — gene-protein-reaction (GPR) associations.

Stoichiometric models have proven very useful in the field of metabolic engineering because they can be used to predict metabolic fluxes and how they are affected by genetic manipulations (McCloskey et al., 2013). GEMs can be used with a panoply of different algorithms to guide the design of cell factories: predict the effect of gene/reaction knockouts, gene over and under expression, and insertion of heterologous pathways (Machado and Herrgård, 2015, Maia et al., 2016).

It is also possible to combine GEMs with omics data to improve their predictability (Maia et al., 2016). In every step of the cycle, data from previous iterations can be integrated with the model to generate better predictions. The next generation of these models is already under development. ME-models (metabolism and expression) are an expansion of the GEMs and account for protein expression. These new models include the process of transcription and translation (O'Brien and Palsson, 2015). ME-models are better at predicting genetic intervention targets *in silico* (King et al., 2017).

After the manipulations are identified and prioritized, it is necessary to design the genetic

elements. There are two major consideration when optimization the DNA sequences: available genetic engineering tools and experimental setup. DNA sequences can also be optimized for hosts-specific codon-usage. There are already a good deal of synthetic biology computational tools available to help with these tasks (Marchisio and Stelling, 2009).

Build Phase

In the build phase, the modifications planned in the design phase are introduced to the host. The challenges in this phase fall on three main categories: to overcome physiological limitations, to implement the genetic modifications to the host and to correctly assembly the DNA parts. Molecular biology tools are used to introduce the genetic manipulations and heterologous elements, while evolution can help to overcome the physiological limitations of the host (Nielsen and Keasling, 2016).

The raw materials and the final products can be toxic to the host in high concentrations, as well as the intermediate products in the metabolic pathways. Besides, some hosts have latent uptake pathways that are inefficient or not expressed. Adaptive laboratory evolution ALE is a method for phenotypic improvement that consists of selecting the best performing strains (e.g., the fastest growing strain) under selective pressure, such as a new carbon source, different temperature, or presence of toxic compounds. It can be used to improve the tolerance to toxic chemicals or to enable the uptake of chemicals from the medium (Hansen et al., 2017, Lee and Kim, 2015).

The CRIPSR!/Cas9 technology has enabled efficient implementation of multiple genetic modifications (i.e., gene insertions or knockouts (Jakočiūnas et al., 2015), and up or down regulation (Gilbert et al., 2013)) in a single transformation and it works in different hosts.

Inserting complex heterologous pathways is, however, challenging. It is necessary to efficiently assemble parts of DNA together and reach the right balancing of expression levels in order to achieve optimal conversion rates. Modern molecular biology provides a huge number of tools to put the parts together. DNA parts can be assembled *in vivo* or *in vitro*, with different number of elements and sizes(Cavaleiro, 2016, p.14-43). Combinatorial assembly can be used to generate libraries of clones with different expression levels (Smanski et al., 2014).

Test Phase

The performance of strains is evaluated in the test phase. Different aspects of the process are evaluated in this phase: how tolerant has a strain become to a given chemical; how many genes where successfully knocked-out; what is the expression level of up or down regulated genes; are the inserted genes being expressed; which of the heterologous assembly combinations resulted in production of the desired chemicals; and what are the yields, titers and productivity of those chemicals; what are the growth rates of the strains (Lee and Kim, 2015, Nielsen and Keasling, 2016, Petzold et al., 2015).

The target molecules and their amounts can be detected using analytical chemistry assays like mass spectrometry (MS), gas chromatography (GC), liquid chromatography (LC). These assays are very reliable, however, they are not very high-throughput. If the number of clones generated is very large, then high-throughput screening (HTS) is preferred, because the analyzes are time-consuming and expensive (Petzold et al., 2015).

In the past years, high-throughput technology, like Next-Generation Sequencing (NGS) and shotgun proteomics provide quantitative measurements of transcript and protein levels. Such data is useful to verify if the strains have been correctly engineered and to identify possible bottlenecks (Petzold et al., 2015).

Biosensors play an important role in the development of cell factories. They provide HTS without the need of classical analytical methods (i.e. GC, LC or MS), because they allow *in vivo* reporting of chemical intracellular concentrations. Sensors can be combined with selection (e.g., toxic compounds or antibiotics) or sorting techniques (e.g., fluorescence-activated cell sorting (FACS)) to identify the best performing strains (Genee, 2016, p19–24).

Learn Phase

Before moving to the next iteration, it is important to gather some knowledge about the strains developed. The learn phase has not always been considered a step in cell factory development and it still does not hold much emphasis in the entire process (Nielsen and Keasling, 2016). Systematically measuring and storing relevant information about the genotype and the associated phenotype can improve the designs. Nevertheless, pioneering works are emerging using tools like design of experiment (DoE), machine-learning and statistical analysis to provide enlightenment

during the engineering process.

Statistical-based analysis is also showing promising results. Statistical models, such as DoE have been used to combine expression data with mathematical modeling to improve productions (Xu et al., 2017). Proteomics data has also been used in combination with machine-learning to enhance protein production (Sastry et al., 2017).

More recently, a multi-omics framework — combining statistical analysis, machine-learning and GEMs — was applied to production of mevalonate derived products in *E. coli*. The framework has shown how to find useful patterns in the data that were used to improve the production process and to identify genetic intervention targets that lead to higher production (Brunk et al., 2016).

Evolutionary Engineering

Evolutionary engineering can be used as an alternative approach to improve production of metabolites or proteins or other industrially relevant phenotypes such as tolerance to stresses. The evolutionary engineering approach consists of generating genetic variation and applying selection and screening methods to achieve the desired phenotypes (Sauer, 2001). A landmark of evolutionary engineering is the production of penicillin. Starting in the 40's, *Penicillium chrysogenum* went through several rounds of mutagenesis to create strains capable of production high amounts of penicillin (Barreiro et al., 2012). Evolutionary engineering is a practical solution to overcome the complexity of cells in cases where suitable selection or screening methods are available. It can furthermore be used when rDNA technologies are not allowed due to regulations or when genetic engineering tools are not available. Evolutionary engineering has attracted increasing attention recently due to rapid development of DNA sequencing technology allowing the use of whole genome sequencing (WGS) (i.e., to sequence the complete DNA of an organism at once) to identify the mutations in strains obtained using evolutionary engineering approaches.

Genetic variation can be obtained using classic strain improvement (CSI, e.g., chemical, UV or X-Ray mutagenesis) or ALE — exposure to stress conditions (e.g., toxic compounds, high-temperature or alternative carbon sources) in serial or continuous cultivation routines relying primarily on natural mutation (Alkım et al., 2014). *In vivo* recombination (i.e., mating or conju-

gation) also provides the means of increasing genetic variability (Sauer, 2001). The cells able to survive the experimental procedure acquire new phenotypes. In general the screening or selection process identifies mutants with the desired phenotype, but the exact process depends on the application (e.g. (Johansen et al., 2015, Steensels et al., 2014)). Evolutionary engineering is nowadays often used in an iterative fashion in order to obtain improved production strains (Figure 1.4). In the evolutionary engineering cycle metabolic models can be used as tools to interpret genotypic and phenotypic data and to design either improved selection regimes or genetic changes that improve the selection for the desirable phenotype.

Design Phase

In this alternative *design phase* we can plan strategies to achieve the desired phenotype. For example, ALE experiments can be planned according to the desired phenotype: increased tolerance to a given chemical or improved growth on a latent carbon source (Dragosits and Mattanovich, 2013, Hansen et al., 2017). GEMs can be used to plan evolutionary experiments. It has already been shown that *E. coli* evolved strains perform as predicted by these models (Fong et al., 2005, Ibarra et al., 2002) at least in the case of growth on alternate carbon sources.

GEMss also have been used to predict antimetabolite targets for cancer treatment (Agren et al., 2014). This functionality can be extended to design strains in the context of evolutionary engineering. When exposed to toxic chemicals, cells will rearrange their phenotype. The survivors are likely to block the conversion of toxic chemicals (e.g., mutations transports, enzymes or regulatory genes). In this PhD thesis, we propose new a computational method to identify metabolite targets that can be use to reshape the metabolism. With this method we can identify phenotypes that can be achieved by supplying toxic analogues (see chapter 3).

Finally, the actual ALE experiments can be simulated using ALEsim (LaCroix et al., 2017). This software allows the user to simulate different parameters of the ALE experiment such passage size, cell count or duration of the experiment in order to design the experimental protocol.

Mutation and selection

The *mutation and selection* phase consists of inducing mutations in the microorganisms and selecting for the best performing strains. UV and chemical mutagenesis are easy to apply to any

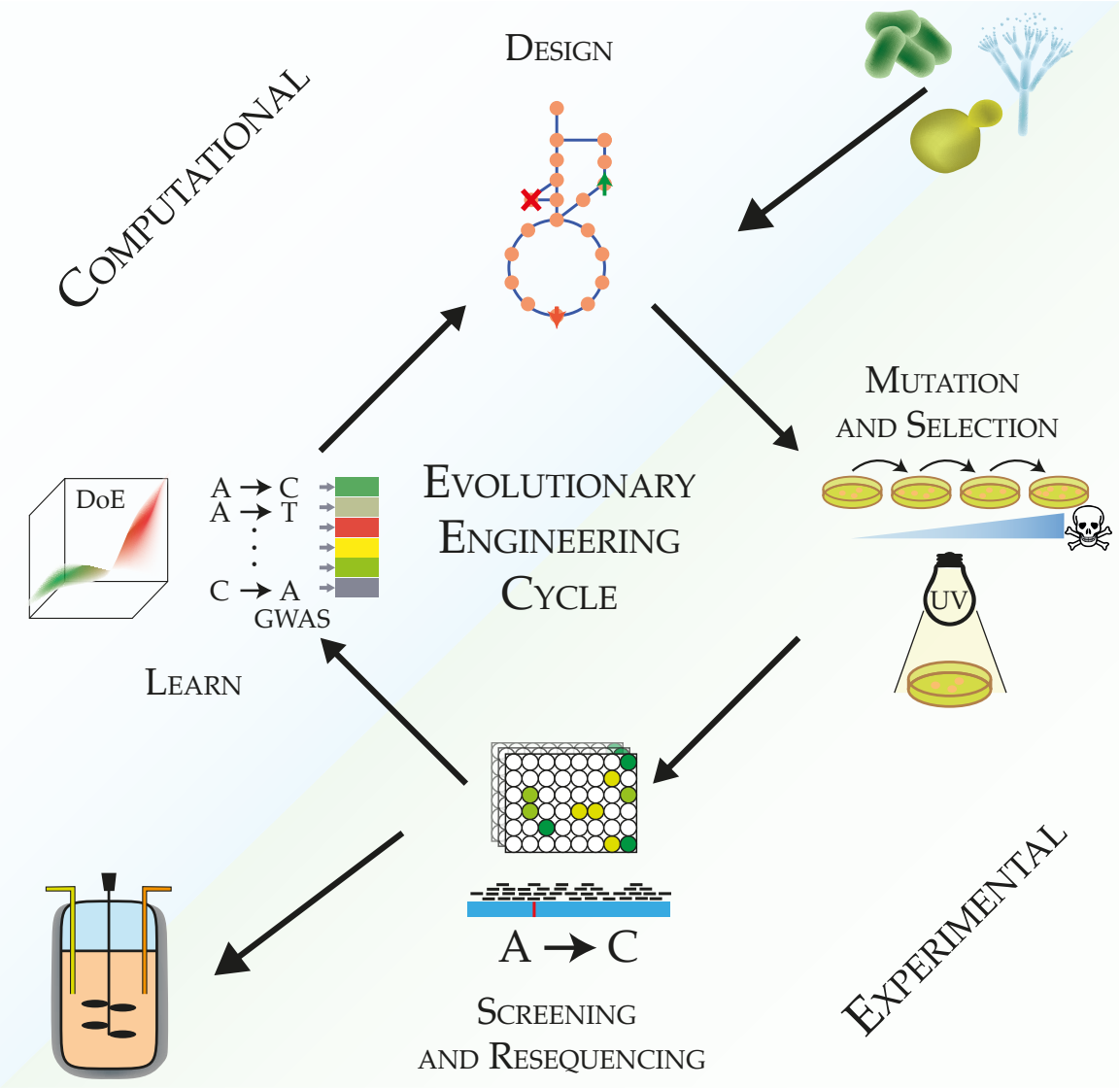


Figure 1.4: Alternative cell factory development cycle using evolutionary engineering. We can use evolution to develop strains with new phenotypes in four steps: *design*: use computation models to predict the expected phenotypes from changing carbon-sources or exposure to toxic chemicals; *build*: use ALE or CSI to evolve the cells using suitable selection pressure; *test*: identify which isolates have the desired phenotype and where the mutations occur in the isolate genomes; and *learn*: analyze the genotype-phenotype data to explain causality and understand the effect of the mutations.

organism. Most mutagenesis strategies are biased either in the types of mutations they induce or locations of these mutations (Sauer, 2001). This reduces the sampling of the potential sequence

space and thus the efficiency of mutagenesis-based strategies.

Exposing cells to toxic non-mutagenic chemicals can yield more understandable results than exposure to mutagens. For example, exposing *Streptococcus thermophilus* to 2-deoxyglucose (a glucose analogue) resulted in mutants that cannot phosphorylate glucose. These mutants secrete glucose instead of consuming it (Sørensen et al., 2016) - a desirable phenotype in the dairy industry. Chemical targets predicted by metabolic models can be implemented the same way.

ALE allows for a more sophisticated selections. Phenotypes that can be coupled to growth (i.e., production of a chemical or utilization of a substrate) can be obtained by selecting the fast growing strains (Hansen et al., 2017). Selections can also be performed in variable conditions in order to optimize cellular responses to environmental changes.

Screening and resequencing

After obtaining isolates after applying the relevant selection, it is necessary to characterize two key features of these isolates: mutations and phenotypes. Identification of mutations can be achieved using WGS. NGS provides a fast, cheap and accurate way to resequence the entire genome and there are severl algorithms available to map different types of mutations as long as a reference wild type reference genome is available (LaCroix et al., 2015). The phenotypes that need to be characterized include growth behavior in the selection or other relevant conditions as well as production of desired metabolites. In most cases standard analytical methods are required to characterize production phenotypes, but in some cases colorimetric or other high throughput assays can be applied.

Learn

Not all the mutations occurring during the mutagenesis are responsible for the desired trait. It is necessary to find causal mutations for phenotypes. Genome-wide association studies GWAS has been used to link genetic traits to features in population (e.g., susceptibility to diseases (Chapman and Hill, 2012) or the linage of the individuals (Rosenberg et al., 2010)). genome-wide assotiation studies (GWAS) has also been used to characterize sequence determinants of bacterial phenotypes such as virulence and resistance (Lees and Bentley, 2016). Similar methods can be used to compute the contribution of single nucleotide variations (SNVs) to the phenotype in the ALE setting,

given that we know which trait strains were selected for and we have a sufficiently large number of isolates resequenced and phenotypically characterized (Jensen, 2015).

Predicting the effect of mutations on phenotypes in general is a challenging task even for bacteria. A major general challenge that needs to be solved to predict mutation effects, is the ability to predict effects of mutations on protein activity. In this thesis, we reviewed the tools available for evaluating the effect of genetic variation and challenges of combining those effects with GEMs (Chapter 5). We also evaluated the possibility of predicting the effects of genetic variation in enzymes (Chapter 6). The initial attempts to bridge genetic variation and GEMs systematically has shown a moderate correlation between predicted and observed growth rates of ALE mutants (Jensen, 2015). The ability to predict effects of mutations on enzyme activities would allow closing the evolutionary engineering loop (Figure 1.4) as GEMs could be modified in a systematic fashion to account for mutations that are present after taking an isolate through the loop.

Computer-aided design using genome-scale metabolic models

The usage of computers to help solve design problems dates back to the 40s and 50s where they were used for designing electrical systems. The term computer-aided design (CAD) appeared a few years later, describing the use of computers to aid in the process of creating, modifying and analyzing designs (Ross, 1960). CAD quickly spread across multiple fields, such as civil engineering and architecture, automobile industry, entertainment and animation, medicine, and many others. Two key features of modern CAD software are the ability to use models to describe the systems under study and to help provide insight into the designs. With no exception, biological sciences have been adopting these tools.

In the field of metabolic engineering, the biological systems we are trying to modify are complex. The models available to describe the metabolism, such as GEMs, kinetic models or ME-models, can be very large and interconnected. For example, the latest GEM for *Escherichia coli* contains 2251 reactions, 1136 metabolites and 1366 genes (Orth et al., 2011). Moreover, the algorithms necessary to run simulations with the models are not trivial either. Identifying and prioritizing the metabolic engineering strategies is a multistep process (Brochado and Patil, 2014).

It requires mathematical programming and the number of results generated can be very large. Working with such models is not an easy task and it cannot be easily achieved using canonical software like Microsoft Excel.

Available software

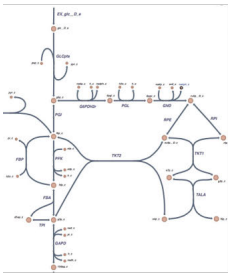
There is CAD software already available for metabolic engineering. Two good examples are OptFlux (Rocha et al., 2010) and the COBRA Toolbox (Hyduke et al., 2011). OptFlux is a standalone application that can be use to perform state-of-the-art optimization of cell factories using GEMs. The COBRA Toolbox is a constraint-based modeling MATLAB library with metabolic engineering capabilities using extension scripts. On one hand, the COBRA Toolbox is more versatile than OptFlux, because it is script-based, so users can code and create their own workflows. On the other hand, OptFlux uses a graphical user interface, which is more friendly for researchers with no programming skills. However, the methods and workflows are predefined and running multiple individual simulations can become tedious. These two software tools are open-source, but the COBRA Toolbox requires MATLAB, which is a commercial software. OptFlux is free to use, but under the restrictive GPLv3 license.

In the next chapter of this thesis we present *computer-aided metabolic engineering and optimization (cameo)*, a CAD software that is open-source, free to use, and user-friendly. We developed *cameo* to tackle major problems: provide a truly open and free to use library for metabolic engineering, create a reference community tool, make modeling accessible to people without engineering and computational background, and easy-to-deploy software in personal computers or high performance computing (HPC) stations.

Constraint-based modeling

In the context of chemical production, GEMs are a powerful and versatile tool. These models can be used to calculate flux distributions by constraining them to reflect different scenarios. Some examples include the effect of gene or reaction knockouts, different media compositions, expression profiles, physiological data (i.e. growth, secretion and uptake rates), metabolite or enzyme concentrations, or presence of new enzymes (Blazier and Papin, 2012, O'Brien et al., 2015, Sánchez et al., 2017, Yizhak et al., 2010).

A GEM can be represented using a $i \times j$ stoichiometric matrix ($\mathbf{S}$) with i rows, representing the set of unique metabolite species present in specific sub-cellular compartments, and j columns, to represent the reactions. Each cell in the matrix is the stoichiometric coefficient of the i-th metabolite in the j-th reaction (Figure 1.5). The $\mathbf{S}$ matrix can be multiplied by the vector of fluxes v to get the change of rate in metabolite concentrations $\Delta[X]/t$.



Reaction	Stoichiometry
TALA	g3p + s7p = e4p + f6p
TKT1	r5p + xu5p-D = g3p + s7p
TKT2	e4p + xu5p-D = f6p + g3p
GND	6pgc + nadp = co2 + nadph + ru5p-D
GAPD	g3p + nad + pi = 13dpg + h + nadh
PGK	3pg + atp = 13dpg + adp
FBA	fdp = dhap + g3p

S =

	TALA	TKT1	TKT2	GND	GAPD	PGK	FBA
e4p	1	0	−1	0	0	0	0
co2	0	0	0	1	0	0	0
f6p	1	0	1	0	0	0	0
nadh	0	0	0	0	1	0	0
s7p	−1	1	0	0	0	0	0
nadp	0	0	0	−1	0	0	0
13dpg	0	0	0	0	1	1	0
r5p	0	−1	0	0	0	0	0
h	0	0	0	0	1	0	0
pi	0	0	0	0	−1	0	0
ru5p	0	0	0	1	0	0	0
nadph	0	0	0	1	0	0	0
dhap	0	0	0	0	0	0	1
g3p	−1	1	1	0	−1	0	1
nad	0	0	0	0	−1	0	0
adp	0	0	0	0	0	1	0
atp	0	0	0	0	0	−1	0
6pgc	0	0	0	−1	0	0	0
xu5p	0	−1	−1	0	0	0	0
3pg	0	0	0	0	0	−1	0
fdp	0	0	0	0	0	0	−1

Figure 1.5: How to build a stoichiometric matrix. A metabolic network can be described as a set of equations representing the stoichiometry of each reaction. To build the matrix we simply iterate through every metabolite and reaction and set the coefficient of the metabolite for that reaction. If the metabolite is consumed by a reaction, then the coefficient is negative, and positive if it is produced. If the metabolite does not participate in that reaction, the coefficient is zero.

If we assume a pseudo steady-state, where the concentrations of metabolites remain unchanged, then $\mathbf{S} \cdot \mathbf{v} = 0$. From here, we can transform this matrix into a system of linear equations (Figure 1.6B-C). However, this matrix usually has numerous degrees of freedom, which makes the system underdetermined. Therefore, there are multiple solutions that satisfy this system and it cannot be solved analytically without determining the value of some variables.

Given the structure of the model — which is imposed by the mass balance of the reactions — we can optimize relevant objective functions (e.g., maximize growth rate) using linear programming (LP) by formulating the problem described in Figure 1.6D, where v_{lb} and v_{ub} define the limits of each flux. c is a vector of zeros, except for the reactions to maximize. Because of the mass balance, constraining the uptake rate of carbon is enough to bound the system. The growth rate can be determined by maximizing the biomass reaction — a reaction describing the stoichiometric composition of metabolites present in one gram of biomass. In this case, c will be filled with zeros, except at the index of the biomass reaction. This method is called Flux Balance Analysis (FBA) (Orth et al., 2010) (Figure 1.6).

Basic operations using GEMs

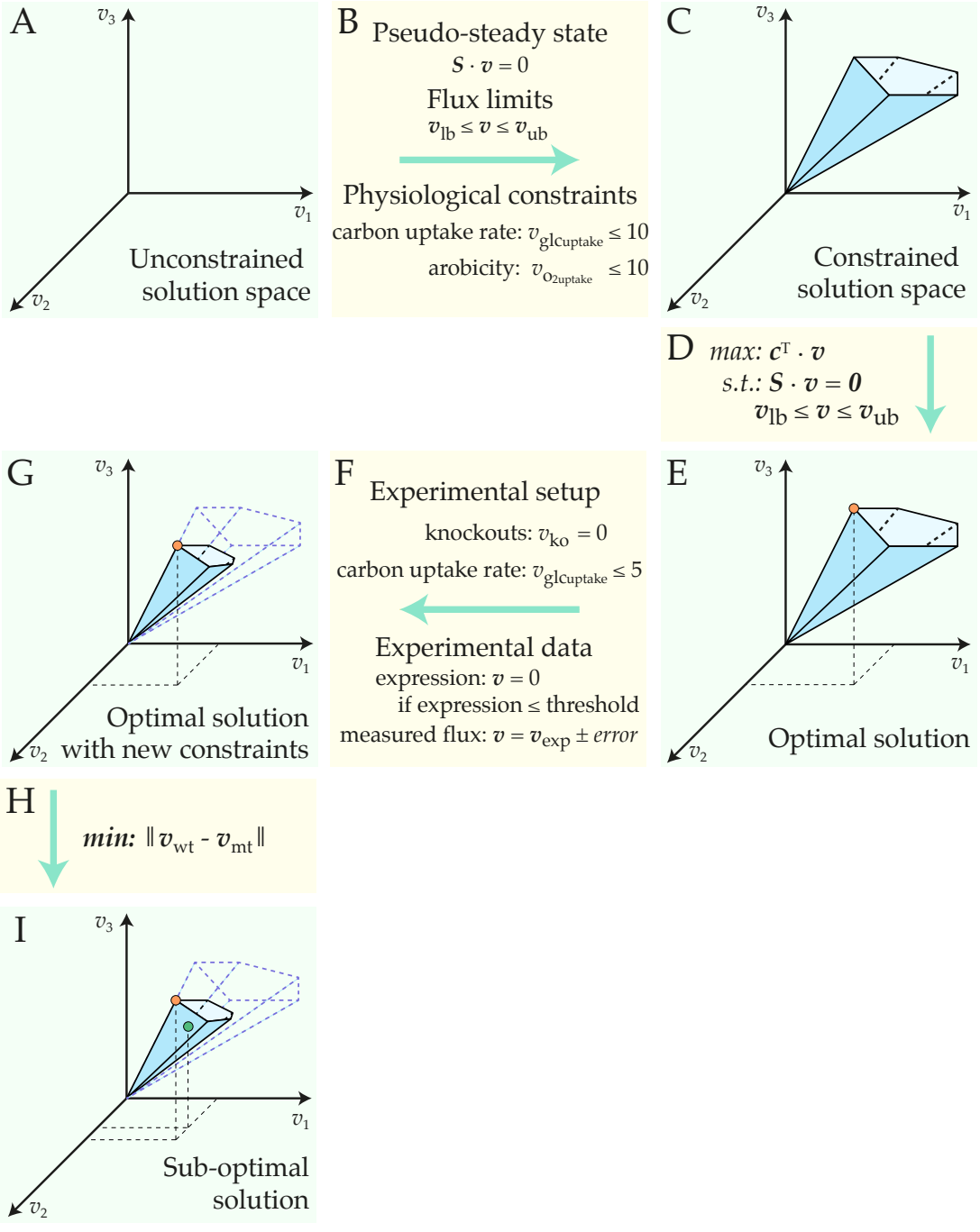
The basic operations performed with a GEM consist of changing the reaction bounds and the coefficients of the stoichiometric matrix. These tasks are essential to test hypothetical scenarios: knockout effects, different media composition and changing the cofactor affinities of enzymes. A good CAD should simultaneously provide an easy and intuitive way to execute those tasks and validation, such as checking mass-balance and charge balance or identification of infeasible constraints and bounds.

Reaction knockouts can be tested by setting the lower and upper bounds to 0 (Figure 1.6F). However, the reactions are catalyzed by enzymes encoded in the genome. Therefore, it is not possible to knockout actual reactions, but genes encoding the enzymes that catalyze them. The relationship between genes and reactions is not always one-to-one. We use GPR rules to describe which genes are responsible for each reaction. These association rules are encoded using Boolean logic (1.7). The effect of knocking out genes can be assessed by testing the GPR rule of each reaction. If the minimal number of genes necessary to express the reaction are knocked-out, then the lower and upper bounds of the reactions are set to 0.

Knockout strains commonly exhibit sub-optimal phenotype (i.e., lower growth and production rates). To achieve the phenotype predicted with FBA, they need to adapt (Segrè et al., 2002). To accommodate the missing activities, other internal regulatory modifications are required to reroute the metabolic fluxes. The changes necessary to recover the optimal phenotype (i.e., maximum growth rate) will not be observed when the cells are cultivated. Instead, we will likely observe a sub-optimal phenotype (Segrè et al., 2002, Shlomi et al., 2005).

Three methods have been proposed to account for the effects of regulatory reprogramming. The Minimization of Metabolic Adjustment (MOMA) (and a linear derivation, Linear Minimization of Metabolic Adjustment (lMOMA)) assume that the cells will try to retain the previous fluxes, given the possible available routes. The new mathematical objective becomes min :

Figure 1.6 *(following page)*: Consequences of mathematical constraints imposed by the genome-scale metabolic model. Given a hyperspace where the reaction fluxes exist (A), the pseudo-steady state assumption and environmental uptake rates (B) define a hypercone (C). Because the hypercone is convex, we can use mathematical programming (D) to identify optimal solutions in inside the cone (E). Other constraints — explaining genetic interventions, experimental conditions, and measurements (F) — can be used to constrain the space further (G). Microorganisms need time to adjust to the genetic modifications (H) resulting sub-optimal flux distribution (I).


A
v_3
v_1
v_2
Unconstrained solution space
B
Pseudo-steady state
$S \cdot v = 0$
Flux limits
$v_{lb} \le v \le v_{ub}$
Physiological constraints
carbon uptake rate: $v_{glcuptake} \le 10$
arobicity: $v_{O2uptake} \le 10$
C
v_3
v_1
v_2
Constrained solution space
D
$max: c^T \cdot v$
$s.t.: S \cdot v = 0$
$v_{lb} \le v \le v_{ub}$
E
v_3
v_1
v_2
Optimal solution
F
Experimental setup
knockouts: $v_{ko} = 0$
carbon uptake rate: $v_{glcuptake} \le 5$
Experimental data
expression: $v = 0$
if expression ≤ threshold
measured flux: $v = v_{exp} \pm error$
G
v_3
v_1
v_2
Optimal solution with new constraints
H
$min: \| v_{wt} - v_{mt} \|$
I
v_3
v_1
v_2
Sub-optimal solution

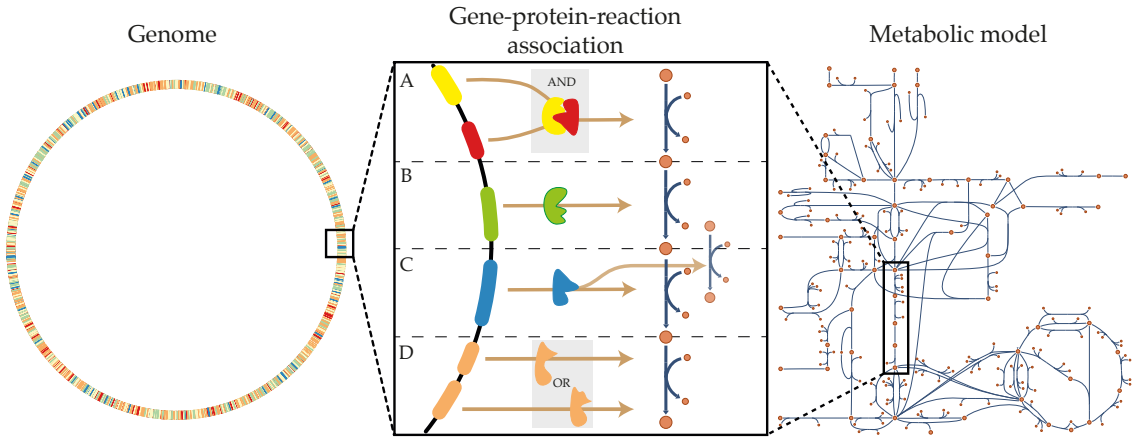


Figure 1.7: Gene-protein-reaction GPR associations. The bridge between the metabolic network (on the left) and the genome (on the right) in aGEM is the GPR association. There are three possible combinations of enzyme cases: A) multiple peptides - one protein (protein complexes); and B and C) one peptide - one protein; D) different peptides - similar protein (isozymes). In some cases (e.g, case C), the same enzyme can catalyze different reactions.

$\|v - v_{wt}\|^2$. This formulation minimizes the euclidean distance between the parent strain (v_{wt}) and the mutant strain (Segrè et al., 2002). Computing the euclidean distance is quadratic programming (QP) problem, which is computationally intense comparing with LP. The mathematical formulation can be converted to the 1-norm (i.e., Manhattan distance) and it becomes linear.

An alternative way to address this problem is using mixed-integer linear programming (MILP). The cells will need to express new proteins to activate fluxes. For that reason, we can constrain the number of new activities in the cell and keep using the existing ones. The Regulation on/off Minimization (ROOM) method implements that assumption (Shlomi et al., 2005).

The Minimization of Metabolites Balance (MiMBl) approach overcomes the effect of the stoichiometric representation. The flux values depend on the stoichiometric coefficient. This approach introduces the metabolite turnover, that is the sum of all fluxes producing a metabolite. By minimizing the change in turnover followed by the minimal change in fluxes that support it, this new approach shown improved sensitivity comparing with the previous methods (Brochado et al., 2012).

All approaches have proven to be better than FBA at predicting the phenotype of cells after single gene deletion in laboratory strains. MiMBl also performs well with multiple knockouts, being able to capture epistatic interactions.

Changing media can be achieved by changing the bounds of boundary reactions. Boundary reactions represent metabolites that come in and out of the system. By changing the bounds of those reactions, we can change the available carbon, nitrogen, oxygen, sulfur and phosphate sources, as well as micro-nutrients (e.g., iron, magnesium, calcium, etc.).

Swapping cofactors is done by replacing the coefficients of metabolites in the stoichiometric matrix (Figure 1.8. For example, to test if producing more NADPH instead of NADH can be used to improve a production strain. It has been hypothesized that changing the redox balance in *S. cerevisiae* could make this host more flexible for production of chemicals (Rossum et al., 2016b).

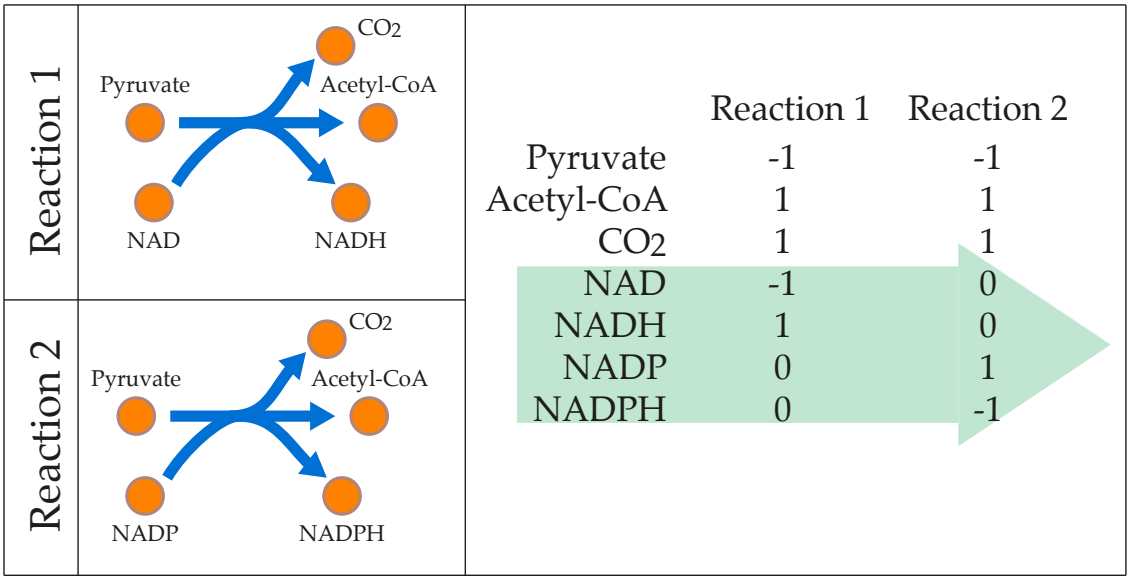


Figure 1.8: Swapping cofactors. The two reactions can be carried with different cofactor (NAD and NADH or NADP and NADPH). To change the cofactor in the model, we simply need to change the stoichiometric coefficients of the cofactors in the reaction in the stoichiometric matrix. Changing just one of the cofactors is against the mass-balance principles and render the model invalid.

Manipulating the model is required for the implementation of workflows, such as solution space analysis (e.g., flux variability analysis (Mahadevan and Schilling, 2003) or production envelope) and identification of essential genes, reactions and metabolites (Edwards and Palsson, 2000, Kim et al., 2007).

Solution space analysis

Analysis of the solution space can be used to inform about the impact of changing constraints in the model. Media conditions, specific growth and production rates, knockouts and other constraints will have an impact on the flux distribution .

Essential reactions are those reactions required to support a given cellular objective (e.g., growth or ATP production). They can be found by individually knocking out each reaction and testing if the model objective still can be computed. Likewise, *essential genes* can be identified using the same iterative approach (Edwards and Palsson, 2000). *Essential metabolites* are defined as metabolites necessary to support the cellular objective. They can be identified by blocking the production of each metabolite . In order to allow feasible results, the mass balance constraint for that metabolite should be relaxed in order to allow its production (Kim et al., 2007)

The optimal solution of FBA is not always unique. There are different combinations of fluxes that can result in the same flux distribution. The existence of these alternative optimal solutions depends on the model constraints (i.e., medium composition and reaction knockouts). We can use Flux Variability Analysis (FVA) to explore the solution space. The method consists of minimizing and maximizing the flux of each reaction while constraining the previous objective (e.g., maximize growth) to its previous solutions (Figure 1.9C).

Heterologous pathway prediction

Identification of heterologous pathways is not an easy task. There are currently 43269 unique reactions identified in MetaNetX (version 3.0, downloaded 15th August 2017). Identification of smaller linear pathways that depend on one precursor can easily be put together by the human brain. However, identifying long pathways with multiple precursors is a large combinatorial problem.

The combination of GEMs, a database of biochemical reactions, and mathematical programming allows the identification of the shortest possible pathways [OptStrain, (Pharkya et al., 2004)]. Enumeration of these pathways can done using MILP. A binary variable $\mathbf{y}$ can be set on or off based on the flux carried by a reaction:

$$\mathbf{v}_{lb} \cdot \mathbf{y} \leq \mathbf{v} \leq \mathbf{v}_{ub} \cdot \mathbf{y}$$

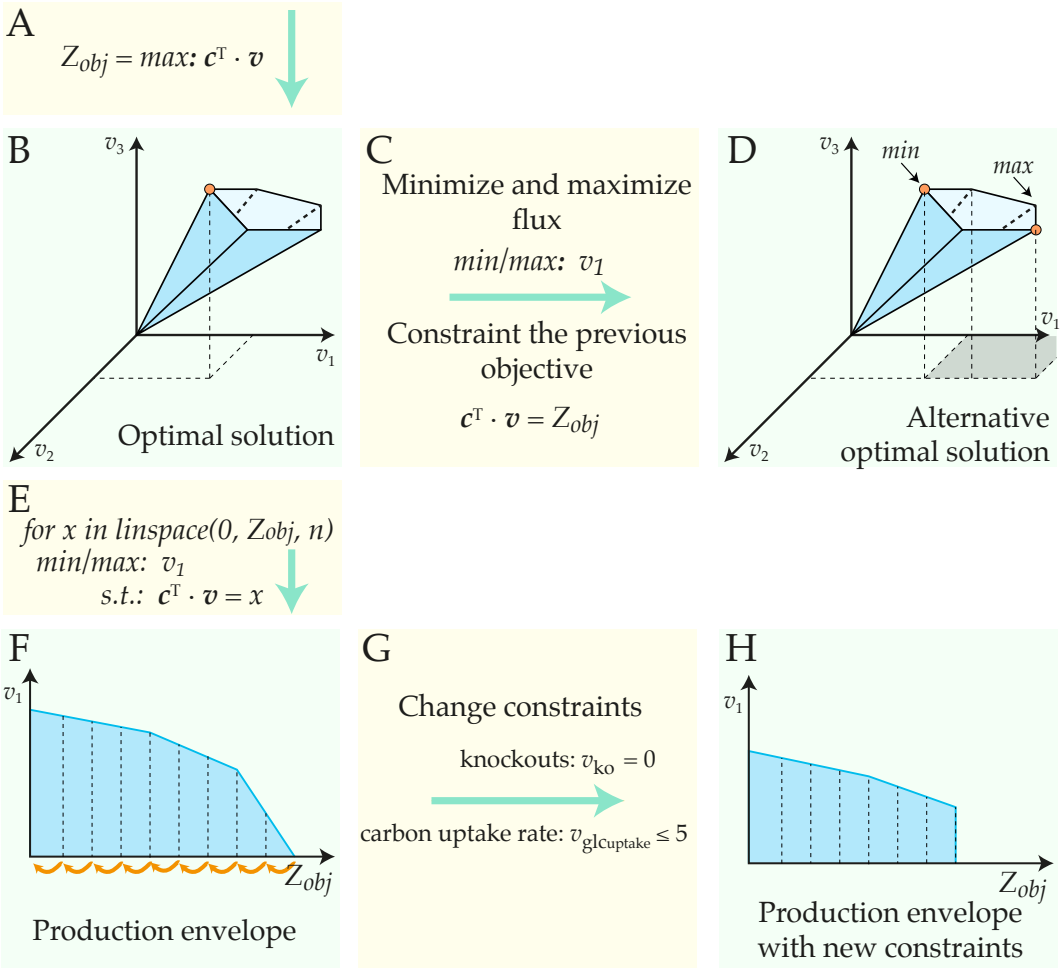


Figure 1.9: Applications of Flux Variability Analysis. FVA can be used to analyze alternative optimal solutions given any cellular objective. By iteratively reducing the objective value, we can explore the flux limits of different reaction (e.g., maximum and minimum flux capacity at different growth rates)

So, for every reaction in a database (consisting of the reactions not present in the GEM) the shortest pathway can be identified by minimizing the sum of $\mathbf{y}$. To do so, we need to extend the stoichiometric matrix with the reactions present in the database.

This can be achieved using mathematical optimization. Considering the set of substrates R (a subset of the metabolites M) that can be consumed by a microorganism, the maximum theoretical yield of a product p can be calculated using LP by solving the following problem:

$$
\begin{aligned}
\text{max:} & MW_p \cdot S_p \cdot \mathbf{v} \\
\text{s.t:} & \sum S_i \cdot \mathbf{v} \geq 0, \forall i \in M \wedge i \notin R && \text{(1.1)} \\
& \sum_{i \in R} (MW_i \cdot S_i \cdot \mathbf{v}) = -1 && \text{(1.2)}
\end{aligned}
$$

where the **MW** is the vector of molecular weights for each metabolite. Constraint 1.1 allows any metabolite to be secreted and constraint 1.2 is used to scale the uptake rate of the substrates to one unit of mass.

The smallest number of necessary reactions to produce a given product p, can be identified by combining the binary variables and the previous problem. Using MILP, we can identify the minimum number of $\mathbf{y}$ variables using the following mathematical problem:

$$
\begin{aligned}
\text{min} : \quad & \|\mathbf{y}\|_1 \\
\text{s.t} : \quad & \textstyle\sum S_i \cdot \mathbf{v} \geq 0, \forall i \in M \wedge i \notin R \\
& \textstyle\sum_{i \in R} (MW_i \cdot S_i \cdot \mathbf{v}) = -1 \\
& MW_p \cdot S_p \cdot \mathbf{v} \geq Yield_p \\
& \mathbf{v}_{lb} \cdot \mathbf{y} \leq \mathbf{v} \leq \mathbf{v}_{ub} \cdot \mathbf{y} \\
y_i \in \{0,1\}, & \forall i \in Universal\ database \\
y_i = 0, & \forall i \in Model
\end{aligned}
$$

Finally, we impose a minimum yield to ensure production of the target. This problem can be implemented in an iterative algorithm to identify alternative solutions. To do that, we can add the following constraint after every iteration, k:

$$
\mathbf{y}_k^T \cdot \mathbf{y} \leq \|\mathbf{y}\|_1 - 1 \qquad \text{(1.3)}
$$

In every iteration, the previous combination of $\mathbf{y}$ cannot be repeated, resulting in an alternative pathway. We can simplify this formulation by simply enforcing a minimum flux value on the target compound.

The enumeration of metabolic engineering targets can be achieved by using four different groups of methods: 1 – mathematical programming; 2 – evolutionary algorithms; 3 – analysis of the solution space; and 4 – elementary flux modes. Groups 1, 2 and 4 are particularly good at identifying growth-coupled designs. A design is growth-coupled when it is not possible to sustain growth without producing the target chemical (Figure 1.10). The coupling strength depends on the mechanism that couples cellular growth with the production pathway (e.g., mass-balance or redox balance).

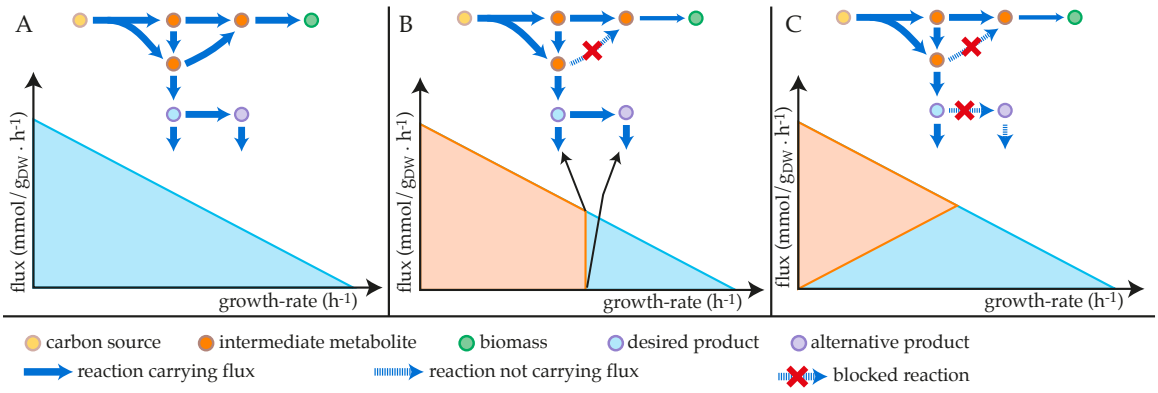


Figure 1.10: Growth-coupled designs explained. Using the network drawn on the top of the figure, A — no constraints are applied — the solution space for different biomass and product fluxes can be any value in the whole stoichiometric space (blue area) . In the B case, the reaction knockout forces the flux to go towards the product pathway. However, with the alternative product being produced from the same pathway, there are multiple possible flux values for the production of the target chemical (orange area). Only when the second knockout is introduced (C), the flux has to go towards the desired product if there is biomass production (orange area).

Group 1: Plain mathematical programming. Using bi-level optimization, a MILP can be defined to identify reaction knockouts that redirect the flux towards a desired target. The problem can be formulated as:

$$
\begin{aligned}
&\max_{\mathbf{y}}: && v_{chemical} && (\textbf{OptKnock})\\
&\text{s.t.} && \left[\begin{array}{ll}
\max: & \mathbf{c}^T \cdot \mathbf{v} \\
\text{s.t.} & \mathbf{S} \cdot \mathbf{v} = 0 \\
& v_{biomass} \geq v_{min\_biomass} \\
& \mathbf{v}_{lb} \cdot \mathbf{y} \leq \mathbf{v} \leq \mathbf{v}_{ub} \cdot \mathbf{y}
\end{array}\right] && (\textbf{Primal})\\
&&& \mathbf{y} = \{0, 1\} \\
&&& \|\mathbf{y}\|_1 \leq K
\end{aligned}
$$

where, the $\mathbf{y}$ variables are binary indicators. If $\mathbf{y}$ equals 0, there is no flux through the reaction. This formulation is called OptKnock and it requires that some constraints are imposed on the model (a minimum growth ($v_{min\_biomass}$) rate, the number of knockouts K). The primal problem (in this specific case FBA) could be any LP (such as lMOMA) (Burgard et al., 2003). Like OptStrain, OptKnock can be used to enumerate knockout strategies using constraint 1.3.

More advanced mathematical formulations have been developed to extend OptKnock. One example is RobustKnock, that ensures growth-coupled designs (Tepper and Shlomi, 2010) by eliminating competing pathways. Another one is OptORF, that accounts for regulatory effects and allows the identification of gene knockouts, transcription factor knockouts and gene overexpression (Kim and Reed, 2010). OptSwap extends RobustKnock and also allows the identification of optimal cofactor swaps to find growth-coupled designs (King and Feist, 2013).

Group 2: genetic algorithms (GAs). These algorithms are inspired by Darwinian evolution. In every iteration of the algorithm the fittest members of a population are selected and reproduce (i.e., generate new individuals). During reproduction, crossover and mutations happen in the individuals chromosomes. The GA evolves towards a global maximum. It is not guaranteed that it will converge to a global optimum like OptKnock.

However, GA algorithms can be used to search over a large number of knockout combinations very fast and allow the optimization objective to be non-linear, providing a broader range of applications (Patil et al., 2005)). Figure 1.11 describes the implementation of a GA for identification of gene knockouts, such as OptGene.

Over- and under-expression can also be computed using GA. This can be achieved by using a discrete representation of gene expression values Gonçalves et al. (2012). There are other variations of these algorithms, also inspired in nature, such as Bee Search (Choon et al., 2012) and differential evolution (Choon et al., 2014). We implemented a version of OptSwap in *cameo* that uses GA .

Group 3 Analysis of solution space. These approaches have two advantages over knockout-based strategies: they can be used to search for non-growth coupled solutions (i.e., chemical compounds that compete with biomass production, such as amino-acids) and they can provide designs with higher yields. The implementation consists of scanning the production envelope and analyzing the flux distributions obtained at each point (Figure 1.12A).

The flux variability scanning based on enforced objective flux (FVSEOF) method analyses the trend of the flux limits for selected reaction groups (1.12B, Trend Analysis). FVSEOF was developed to search for reactions with increased flux, but decreasing patterns can also be identified. Another method is DifferentialFVA, that compares the gap between the flux limits at each point. For a given reaction, if there is a gap between the flux limits at two different points, then the flux needs to increase or decrease (Figure 1.12B, Gap Analysis).

Group 4: Elementary flux mode analysis EFMA. The EFMA can be used to identify all the minimal functional pathways that connect substrates to their products (i.e., biomass and by-products). An elementary flux mode (EFM) is minimal set of reactions. If any reaction in the EFM is blocked, then the EFM cannot carry flux (Zanghellini et al., 2013).

The counterpart of EFMs are the minimal cut sets (MCS). A MCS can be described as the minimal set of reactions that blocks a certain objective. All the reactions in a MCS target all the EFMs that carry flux toward an objective (von Kamp and Klamt, 2014). The enumeration of MCS that can be used to identify genetic engineering targets and to analyze synthetic lethal knockout combinations.

Integration of omics data

GEMs provide a good framework for integration of high-throughput omics data. The data can be used to impose constraints on the model and generate flux distributions that are closer to the cells behavior. There are methods to integrate transcriptomics data (Blazier and Papin, 2012) and enzyme and metabolite concentrations can also be used to constrain the model (Yizhak et al., 2010). However, there is still missing a systematic integration method to account for genetic variation (Chapter 5).

Fluxomics is the measure of flux carried by metabolic reactions. These measurements can be used to constrain the model directly (Figure 1.6F).

Transcriptomics data is the transcript levels, which can be obtained with RNA-Seq. Some methods have been developed to constrain the amount of transcript with the GEMs (Blazier and Papin, 2012, Kim and Lun, 2014).

The methods available try to infer which reactions are on or off using transcript levels. Some methods use hard constraints — block reactions with transcript levels bellow a certain threshold (1.6F). Other, impose soft constraints, by minimizing the inconsistency between the expression data and the fluxes (Becker and Palsson, 2008) or maximizing the number of reactions that are active and expressed (Zur et al., 2010).

The main challenges are: transcript levels have a weak correlation with metabolic fluxes (Moxley et al., 2009) and the relationship between genes and reactions is not one-to-one.

Proteomics data describe the abundance of peptides in the cells. It can be used like transcriptomics data, but we can go beyond that. When combined with conversion rates ($k_{cat}s$), proteomics data can be used to constraint the upper bound of the reactions and generate more accurate flux distributions (Sánchez et al., 2017).

Metabolomics can be used to constraint the model using thermodynamics. Given the free energy of formation of each metabolite and its concentration, it is possible to constraint the direction of the reactions in the GEM (Henry et al., 2007).

Combined omics data can also be used to impose extra constraints on GEMs. For example, gene inactivation moderated by metabolism, metabolomics and expression (GIM3E) combines transcriptomics and metabolomics data. That is achieved by explicitly representing the metabolite turnover as variables and to force the model to produce a very small amount of the detected

metabolites. The transcription data is used to softly constrain the model and a penalty is given to reactions that are required but not expressed (Schmidt et al., 2013).

Integrative omics-metabolic analysis IOMA combines proteomics and metabolics. This is achieved by constraining the flux levels to a Michaelis Menten-like rate equation that can be calculated from protein and metabolite concentrations (Yizhak et al., 2010).

Visualization

Vision is the most important sense in human-computer interaction. Through our eyes we perceive most of the information returned from the computer. The results of data analysis are usually presented in figures, plots, charts and diagrams made to help us understand the data intuitively.

Metabolic networks can be represented as maps. Each map represents a pathway or combination of pathways, connecting metabolic precursors to their products. These maps, contain the possible biochemical routes and represent our understanding of the biochemical conversions inside the cells. Several tools are available for visualization of metabolic networks, both desktop applications and web applications (King et al., 2015). These networks can be used to display data related with the reactions (e.g, fluxes) and metabolites (e.g., concentrations). More data types, such as expression data can also be included, however it requires some assumptions about the relationship between transcript levels and metabolic fluxes.

There are other ways of displaying the data created using the algorithms described before. Production envelopes, show the upper and lower limits of the production flux as function of the growth rate. The shape of the production envelopes can be used to visually compare designs: e.g., which designs have higher yield, or if any of the designs are growth coupled.

Conclusions and perspectives

Biotechnology provided mankind with tools and solutions that led to modern society. The CRISPR/Cas9 technology marks the beginning of a new era. The cost of synthetic DNA has decreased and CRISPR provides a plug and play genetic toolkit that works across many species. Manipulating organisms became a cheap and fast process.

Metabolic engineering has the potential to overcome challenges that mankind will face in

the near future. Sustainable producing of chemicals is still a challenging and expensive process. The duration and cost of building strains will decrease with the help of better technology. The cut in DNA synthesis price and the discovery of CRISPR/Cas9 technology helped decrease the cost of cell factory development (Kosuri and Church, 2014, Sander and Joung, 2014). Biosensors promise a cheap and scalable toolbox to speed up the process, especially in the early iterations of the metabolic engineering cycle. New technologies, like microfluidics, can increase the throughput of built and tested strains in the laboratory and decrease the costs of reagents (Ma and Huo, 2016).

Evolutionary engineering is an alternative process to metabolic engineering. Cell populations under selective pressure can change their regulatory and metabolic traits by changing their genotype. There are three main advantages over metabolic engineering: no need for *a priori* knowledge about the host regulation and metabolism, GMO compatibility, and no need for genetic engineering tools (Derkx et al., 2014, Sauer, 2001). Still, a good screening capacity is required to implement evolutionary engineering.

Evolutionary and metabolic engineering can also be complementary tools to create strains. For example, strains can be evolved to acquire physiological properties (tolerance to products and substrates, high yields) and then engineered to acquire production traits (heterologous pathways or feedback bypass) (Herrgård and Panagiotou, 2012, Sauer, 2001). But it also works the other way around. We can first build our desired strain using metabolic engineering and use evolution to improve the phenotype (Fong et al., 2005, Hansen et al., 2017).

Still, more focus is needed on the design and learn phases. The amount of data generated is increasing and requires more sophisticated analysis methods, capable of integrating different types of data, therefore metabolic engineering needs better software and information management systems.

Groundbreaking initiatives have shown promising results. Unsupervised learning can be used to identify different stages of the fermentation process and pattern recognition can be used to identify relevant changes in the metabolism (Brunk et al., 2016).

Despite the number of algorithms available, modeling of individual processes is still limited approach in metabolic engineering, because they fail to capture the complexity of events happening inside a cell. In the future, integrated models that predict different aspects of the engineering process (e.g., protein expression, protein folding, gene expression, fluxes levels temporal events)

will help increasing the success rate of metabolic engineering (Lechner et al., 2016).

Still, the analysis of genetic variability is not a systematic process and requires effort. The integration with GEMs as well as a generic mutation-trait analysis approaches (i.e., GWAS) can be an effective way of identifying and prioritizing relevant mutations.

Mathematical models describing biological systems are the underlying technology of CAD software for biological sciences. In the context of metabolic engineering, GEMs shown reasonably good accuracy in providing non-trivial solutions to improve chemical production (King et al., 2017, O'Brien et al., 2015). It has been shown recently that the capacity to produce more protein limits the growth rate of *S. cerevisiae* and explains the overflow effect. These can be recapitulated using the GEM extended with enzyme activity constraints (Sánchez et al., 2017). The next generation of models, the ME-models, brings new dimensions the model including protein expression (O'Brien and Palsson, 2015) and are more reliable at predicting cellular phenotypes (King et al., 2017). Many algorithms have been proposed during the last two decades to search for genetic engineering targets and to analyze the data resulting from experiments (Maia et al., 2016).

There are some limitations in the existing software. The available software works, but it requires people with a very specific training to retrieve meaningful results. Moreover, most of the available software is academic, developed by scientists (usually PhD students) and not by software developers. There is a need to create better CAD software that is efficient, reliable, intuitive and user-friendly.

Metabolic and evolutionary engineering should be about making better strains. It is not about knowing all possible details about complex models and algorithms. Retrieving meaningful results requires solid background knowledge about mathematical models and optimization algorithms to overcome small practical problems (e.g., floating point numerical instability, numerical precision or infeasibility analysis). First, such problems belong to a different domain of knowledge. Second, the solutions for such problems should be handled by the software itself. Finally, and most important, the software should be easy and intuitive to use.

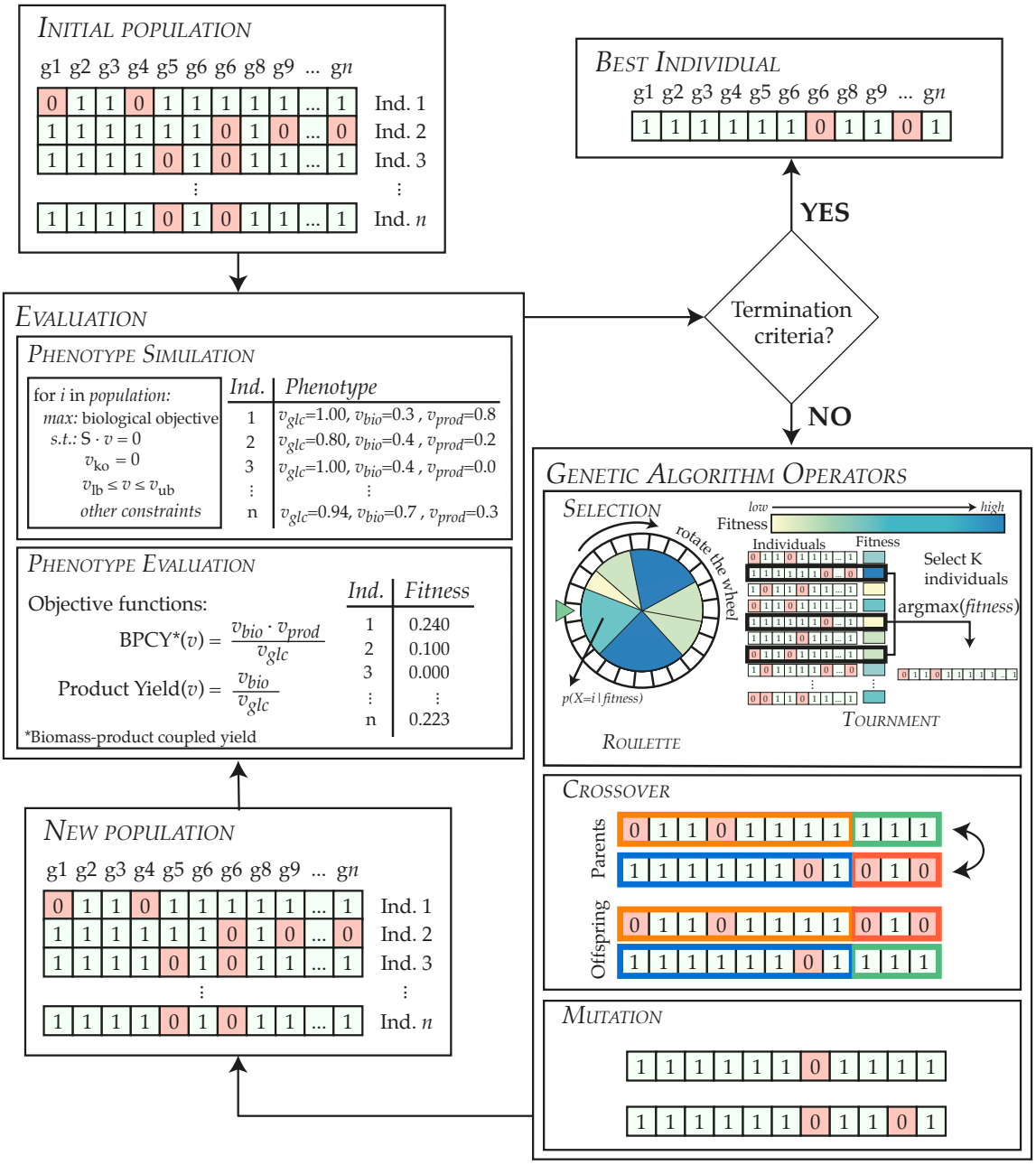


Figure 1.11: Evolutionary algorithm implementation for gene knockout optimization. First, the initial population is randomly generated and evaluated. The algorithm will loop until a termination criteria is met (e.g., number of evaluations, number of iterations, maximum time, or fitness variation). In each iteration, the operator are applied to change the population. The population is re-evaluated in every iteration. When the termination criteria is met, the algorithm returns the fittest individual.

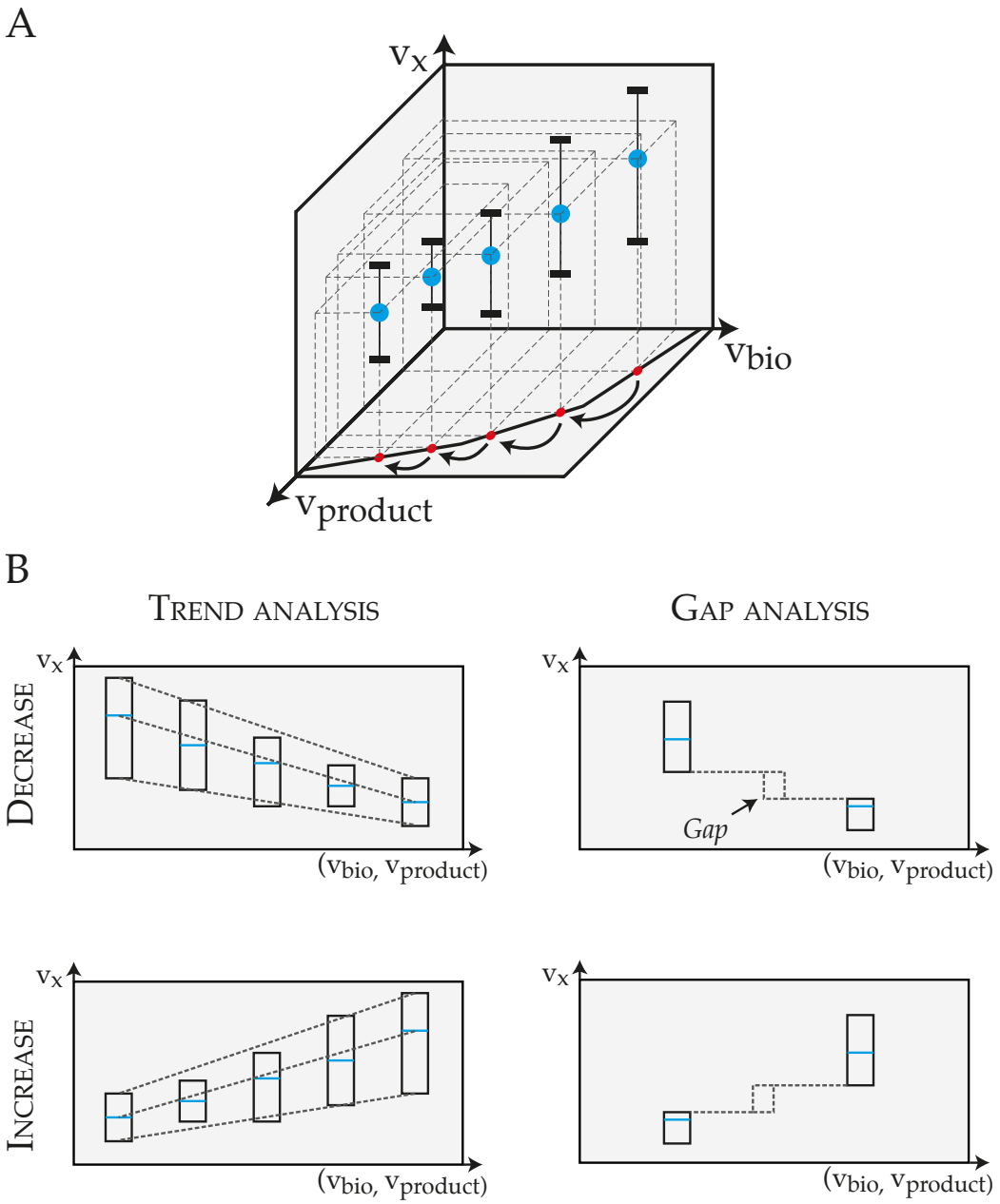


Figure 1.12: Methods based on flux variability analysis. A) Production envelope scan. By fixing specific growth rates and the corresponding maximum theoretical yield, the FVA can be used to calculate the flux limits of specific reactions for different strains. B) Two types of methods can be used to analyze the results: Trend Analysis to find reactions with consistent variation profiles; and Gap Analysis to identify reactions that with fluxes that need to increase or decrease above the reference limits to ensure the desired phenotype.

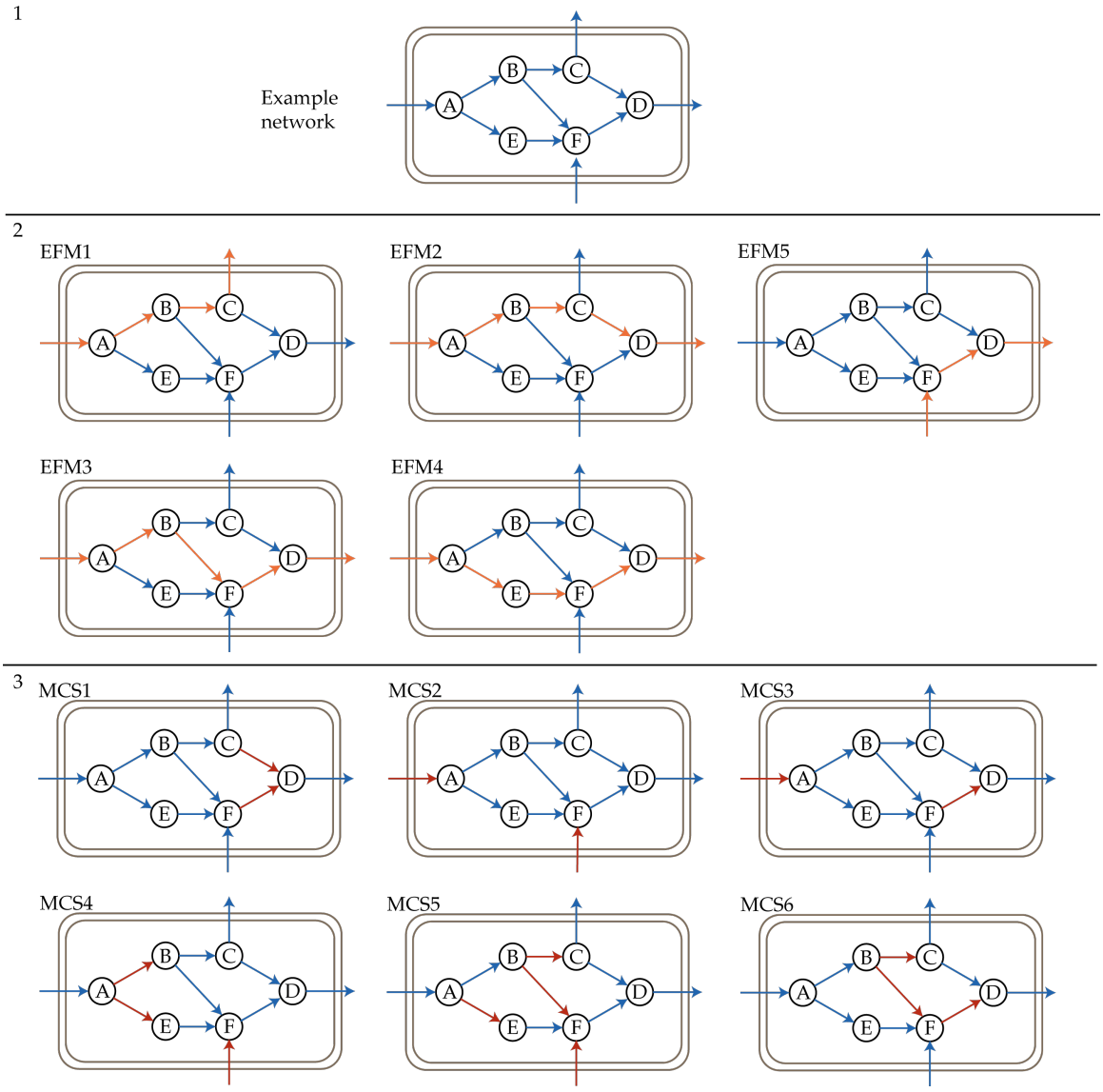


Figure 1.13: Elementary flux modes and minimal cut sets. 1) Sample network with 6 metabolites and 11 reactions. 2) All EFM present in the network. 3) Examples of MCS that block all EFMs that lead to the production of D.

References

Rasmus Agren, A. Mardinoglu, A. Asplund, C. Kampf, M. Uhlen, and Jens Nielsen. Identification of anticancer drugs for hepatocellular carcinoma through personalized genome-scale metabolic modeling. *Molecular Systems Biology*, 10(3):721–721, mar 2014. ISSN 1744-4292. doi: 10.1002/msb.145122.

Ceren Alkım, Burcu Turanlı-yıldız, and Z Petek Çakar. Yeast Metabolic Engineering. In Valeria Mapelli, editor, *Yeast Metabolic Engineering*, volume 1152 of *Methods in Molecular Biology*, chapter 10, pages 169–183. Springer New York, New York, NY, 2014. ISBN 978-1-4939-0562-1. doi: 10.1007/978-1-4939-0563-8.

J E Bailey. Toward a science of metabolic engineering. *Science (New York, NY)*, 252(5013):1668–1675, 1991. doi: 10.1126/science.2047876.

J E Bailey, S Birnbaum, J L Galazzon, C Khosla, and J V Shanks. Strategies and Challenges in Metabolic Engineering. *Annals of the New York Academy of Sciences*, 589(1):1–15, 1990. ISSN 1749-6632. doi: 10.1111/j.1749-6632.1990.tb24230.x.

Carlos Barreiro, Juan F Martín, and Carlos García-Estrada. Proteomics Shows New Faces for the Old Penicillin Producer *Penicillium chrysogenum*. *Journal of Biomedicine and Biotechnology*, 2012:1–15, 2012. doi: 10.1155/2012/105109.

Scott a Becker and Bernhard O Palsson. Context-specific metabolic networks are consistent with experiments. *PLoS computational biology*, 4(5):e1000082, may 2008. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1000082.

Anna S. Blazier and Jason A. Papin. Integration of expression data in genome-scale metabolic network reconstructions. *Frontiers in physiology*, 3:299, 2012. ISSN 1664-042X. doi: 10.3389/fphys.2012.00299.

J E Brandle and P G Telmer. Steviol glycoside biosynthesis. *Phytochemistry*, 68(14):1855–1863, 2007. ISSN 0031-9422. doi: 10.1016/j.phytochem.2007.02.010.

Ana Rita Brochado and Kiran Raosaheb Patil. Model-Guided Identification of Gene Deletion Targets for Metabolic Engineering in Saccharomyces cerevisiae. In Valeria Mapelli, editor, *Yeast Metabolic Engineering: Methods and Protocols*, volume 1152, chapter 10, pages 281—-294. Springer, 2014. ISBN 978-1-4939-0562-1. doi: 10.1007/978-1-4939-0563-8_17.

Ana Rita Brochado, Sergej Andrejev, Costas D Maranas, and Kiran R Patil. Impact of stoichiometry representation on simulation of genotype-phenotype relationships in metabolic networks. *PLoS computational biology*, 8(11):e1002758, jan 2012. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1002758.

Elizabeth Brunk, Kevin W. George, Jorge Alonso-Gutierrez, Mitchell Thompson, Edward Baidoo, George Wang, Christopher J. Petzold, Douglas McCloskey, Jonathan Monk, Laurence Yang, Edward J. O'Brien, Tanveer S. Batth, Hector Garcia Martin, Adam Feist, Paul D. Adams, Jay D. Keasling, Bernhard O. Palsson, Taek Soon Lee, Edward J. O'Brien, Tanveer S. Batth, Hector Garcia Martin, Adam Feist, Paul D. Adams, Jay D. Keasling, Bernhard O. Palsson, and Taek Soon Lee. Characterizing Strain Variation in Engineered E. coli Using a Multi-Omics-Based Workflow. *Cell Systems*, 2(5):335–346, may 2016. ISSN 24054712. doi: 10.1016/j.cels.2016.04.004.

Anthony P. Burgard, Priti Pharkya, and Costas D. Maranas. Optknock: A bilevel programming framework for identifying gene knockout strategies for microbial strain optimization. *Biotechnology and Bioengineering*, 84(6):647–657, dec 2003. ISSN 0006-3592. doi: 10.1002/bit.10803.

Miguel A Campodonico, Barbara A Andrews, Juan A Asenjo, Bernhard O Palsson, and Adam M Feist. Generation of an atlas for commodity chemical production in *Escherichia coli* and a novel pathway prediction algorithm, GEM-Path. *Metabolic engineering*, 25:140–58, sep 2014. ISSN 1096-7184. doi: 10.1016/j.ymben.2014.07.009.

Pablo Carbonell, Anne-Gaëlle Planson, Davide Fichera, and Jean-Loup Faulon. A retrosynthetic biology approach to metabolic pathway design for therapeutic production. *BMC systems biology*, 5(122), 2011. doi: 10.1186/1752-0509-5-122.

Ana M. Cavaleiro. *Improvement of Synthetic Biology Tools for DNA Editing*. PhD thesis, Technical University of Denmark, 2016.

Stephen J. Chapman and Adrian V. S. Hill. Human genetic susceptibility to infectious disease. *Nature Reviews Genetics*, 13(3):175–188, 2012. ISSN 1471-0056. doi: 10.1038/nrg3114.

Yee Wen Choon, Mohd Saberi Mohamad, Safaai Deris, Chuii Khim Chong, Lian En Chai, Zuwairie Ibrahim, and Sigeru Omatu. *Identifying Gene Knockout Strategies Using a Hybrid of Bees Algorithm and Flux Balance Analysis for in Silico Optimization of Microbial Strains*, pages 371–378. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-28765-7. doi: 10.1007/978-3-642-28765-7_44.

Yee Wen Choon, Mohd Saberi Mohamad, Safaai Deris, Rosli Md Illias, Chuii Khim Chong, Lian En Chai, Sigeru Omatu, and Juan Manuel Corchado. Differential bees flux balance analysis with OptKnock for in silico microbial strains optimization. *PLoS ONE*, 9(7), 2014. ISSN 19326203. doi: 10.1371/journal.pone.0102744.

Stanley N Cohen, Annie C Y Chang, Herbert W Boyert, and Robert B Hellingt. Biologically Functional Bacterial Plasmids *In Vitro*. *Proc. Nat. Acad. Sci.*, 70(11):3240–3244, 1973. doi: 10.1073/pnas.70.11.3240.

Chris A Dejong, Gregory M Chen, Haoxin Li, Chad W Johnston, Mclean R Edwards, Philip N Rees, Michael A Skinnider, Andrew L H Webster, and Nathan A Magarvey. Polyketide and nonribosomal peptide retro-biosynthesis and global gene cluster matching. *Nature Chemical Biology*, 12:1007–1014, 2016. doi: 10.1038/nchembio.2188.

Patrick M F Derkx, Thomas Janzen, Kim I Sørensen, Jeffrey E Christensen, Birgitte Stuer-Lauridsen, and Eric Johansen. The art of strain improvement of industrial lactic acid bacteria without the use of recombinant DNA technology. *Microbial cell factories*, 13(Suppl 1):S5, 2014. doi: 10.1186/1475-2859-13-S1-S5.

B S Dien, N N Nichols, and R J Bothast. Recombinant *Escherichia coli* engineered for production of L-lactic acid from hexose and pentose sugars. *Journal of Industrial Microbiology & Biotechnology*, 27(4):259–265, 2001. doi: 10.1038/sj/jim/7000195.

Martin Dragosits and Diethard Mattanovich. Adaptive laboratory evolution – principles and applications for biotechnology. *Microbial Cell Factories*, 12(1):64, 2013. ISSN 1475-2859. doi: 10.1186/1475-2859-12-64.

J S Edwards and B O Palsson. Metabolic flux balance analysis and the in silico analysis of Escherichia coli K-12 gene deletions. *BMC bioinformatics*, 1:1, 2000. ISSN 1471-2105. doi: 10.1186/1471-2105-1-1.

William R. Farmer and James C. Liao. Precursor balancing for metabolic engineering of lycopene production in escherichia coli. *Biotechnology Progress*, 17(1):57–61, 2001. ISSN 1520-6033. doi: 10.1021/bp000137t.

Stephen S Fong, Anthony P Burgard, Christopher D Herring, Eric M Knight, Frederick R Blattner, Costas D Maranas, and Bernhard O Palsson. In silico design and adaptive evolution of Escherichia coli for production of lactic acid. *Biotechnology and bioengineering*, 91(5):643–8, sep 2005. ISSN 0006-3592. doi: 10.1002/bit.20542.

Stephanie Galanie, Kate Thodey, Isis J. Trenchard, Maria Filsinger Interrante, and Christina D Smolke. Complete biosynthesis of opioids in yeast. *Science*, 349(6252):1095–1100, 2015. doi: 10.1126/science.aac9373.

Hans J. Genee. *Towards evolution-guided microbial engineering - tools development and applications*. PhD thesis, Technical University of Denmark, 2016.

Luke A Gilbert, Matthew H Larson, Leonardo Morsut, Zairan Liu, Gloria A Brar, Sandra E Torres, Noam Stern-Ginossar, Onn Brandman, Evan H Whitehead, Jennifer A Doudna, Wendell A Lim, Jonathan S Weissman, and Lei S Qi. CRISPR-Mediated Modular RNA-Guided Regulation of Transcription in Eukaryotes. *Cell*, 154(2):442–451, 2013. ISSN 0092-8674. doi: 10.1016/j.cell.2013.06.044.

Emanuel Gonçalves, Rui Pereira, Isabel Rocha, and Miguel Rocha. Optimization approaches for the in silico discovery of optimal targets for gene over/underexpression. *Journal of Computational Biology*, 19(2):102–14, feb 2012. ISSN 1557-8666. doi: 10.1089/cmb.2011.0265.

Andrey Yu. Gulevich, Alexandra Yu. Skorokhodova, Alexey V. Sukhozhenko, Rustem S. Shakulov, and Vladimir G. Debabov. Metabolic engineering of Escherichia coli for 1-butanol biosynthesis through the inverted aerobic fatty acid β-oxidation pathway. *Biotechnology Letters*, 34(3):463–469, mar 2012. ISSN 0141-5492. doi: 10.1007/s10529-011-0797-z.

G Hames. *Alcohol in World History*. Routledge, 2012. ISBN 978-1108044271.

Anne Sofie Lærke Hansen, Rebecca M. Lennen, Nikolaus Sonnenschein, and Markus J Herrgård. Systems biology solutions for biochemical production challenges. *Current opinion in biotechnology*, 45:85–91, jun 2017. ISSN 1879-0429. doi: 10.1016/j.copbio.2016.11.018.

Esben H Hansen, Birger Lindberg Møller, Gertrud R Kock, Camilla M Bünner, Charlotte Kristensen, Ole R Jensen, Finn T Okkels, Carl E Olsen, Mohammed S Motawia, and Jørgen Hansen. De Novo Biosynthesis of Vanillin in Fission Yeast (*Schizosaccharomyces pombe*) and Baker's Yeast (*Saccharomyces cerevisiae*). *Applied and environmental microbiology*, 75(9):2765–2774, 2009. doi: 10.1128/AEM.02681-08.

V. Hatzimanikatis, C. Li, J. A. Ionita, C. S. Henry, M. D. Jankowski, and L. J. Broadbelt. Exploring the diversity of complex metabolic networks. *Bioinformatics*, 21(8):1603–1609, apr 2005. ISSN 1367-4803. doi: 10.1093/bioinformatics/bti213.

Christopher S Henry, Linda J Broadbelt, and Vassily Hatzimanikatis. Thermodynamics-based metabolic flux analysis. *Biophysical journal*, 92(5):1792–1805, 2007. ISSN 00063495. doi: 10.1529/biophysj.106.093138.

Markus Herrgård and Gianni Panagiotou. Analyzing the genomic variation of microbial cell factories in the era of "New Biotechnology". *Computational and structural biotechnology journal*, 3:e201210012, jan 2012. ISSN 2001-0370. doi: 10.5936/csbj.201210012.

Daniel Hyduke, Jan Schellenberger, Richard Que, Ronan Fleming, Ines Thiele, Jeffery Orth, Adam Feist, Daniel Zielinski, Aarash Bordbar, Nathan Lewis, Sorena Rahmanian, Joseph Kang, and Bernhard Palsson. COBRA Toolbox 2.0. *Protocol Exchange*, 33(1):1–35, may 2011. ISSN 2043-0116. doi: 10.1038/protex.2011.234.

Rafael U. Ibarra, Jeremy S. Edwards, and Bernhard O. Palsson. Escherichia coli K-12 undergoes adaptive evolution to achieve in silico predicted optimal growth. *Nature*, 420(6912):186–189, 2002. ISSN 0028-0836. doi: 10.1038/nature01149.

David A Jackson, Robert H Symonst, and Paul Berg. Biochemical Method for Inserting New Genetic Information into DNA of Simian Virus 40: Circular SV40 DNA Molecules Containing Lambda Phage Genes and the Galactose Operon of *Escherichia coli*. *Proc. Nat. Acad. Sci.*, 69 (10):2904–2909, 1972. doi: doi:10.1073/pnas.69.10.2904.

Tadas Jakočiūnas, Michael K Jensen, and Jay D Keasling. CRISPR / Cas9 advances engineering of microbial cell factories. *Metabolic Engineering*, 34:44–59, 2015. doi: 10.1016/j.ymben.2015.12.003.

Kristian Jensen. Modeling metabolic networks in *E. coli* using genetic variation data. Master's thesis, Technical University of Denmark, 2015.

E. Johansen, G. Øregaard, K.I. Sørensen, and P.M.F. Derkx. Modern approaches for isolation, selection, and improvement of bacterial strains for fermentation applications. In Wilhelm Holzapfel, editor, *Advances in Fermented Foods and Beverages*, chapter 10, pages 227–248. Elsevier, Cambridge, UK, 2015. ISBN 978-1-78242-015-6. doi: 10.1016/B978-1-78242-015-6.00010-4.

Joonhoon Kim and Jennifer L Reed. OptORF: Optimal metabolic and regulatory perturbations for metabolic engineering of microbial strains. *BMC systems biology*, 4:53, 2010. ISSN 1752-0509. doi: 10.1186/1752-0509-4-53.

Min Kyung Kim and Desmond S. Lun. Methods for integration of transcriptomic data in genome-scale metabolic models. *Computational and Structural Biotechnology Journal*, 11(18): 59–65, 2014. ISSN 20010370. doi: 10.1016/j.csbj.2014.08.009.

P.-J. Kim, D.-Y. Lee, T. Y. Kim, K. H. Lee, H. Jeong, S. Y. Lee, and S. Park. Metabolite essentiality elucidates robustness of Escherichia coli metabolism. *Proceedings of the National Academy of Sciences*, 104(34):13638–13642, aug 2007. ISSN 0027-8424. doi: 10.1073/pnas.0703262104. URL http://www.pnas.org/cgi/doi/10.1073/pnas.0703262104.

Stefanie Kind, Steffi Neubauer, Judith Becker, Motonori Yamamoto, Martin Volkert, Gregory von Abendroth, Oskar Zelder, and Christoph Wittmann. From zero to hero – Production of bio-based nylon from renewable resources using engineered *Corynebacterium glutamicum*. *Metab. Eng.*, 25:113–123, 2014. doi: 10.1016/j.ymben.2014.05.007.

Zachary A. King and Adam M. Feist. Optimizing Cofactor Specificity of Oxidoreductase Enzymes for the Generation of Microbial Production Strains—OptSwap. *Industrial Biotechnology*, 9(4):236–246, 2013. ISSN 1550-9087. doi: 10.1089/ind.2013.0005.

Zachary A. King, Andreas Dräger, Ali Ebrahim, Nikolaus Sonnenschein, Nathan E. Lewis, and Bernhard O. Palsson. Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. *PLOS Computational Biology*, 11(8):e1004321, aug 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004321.

Zachary A King, Edward J. O'Brien, Adam M Feist, and Bernhard O Palsson. Literature mining supports a next-generation modeling approach to predict cellular byproduct secretion. *Metabolic Engineering*, 39(August):220–227, jan 2017. ISSN 10967176. doi: 10.1016/j.ymben.2016.12.004.

Sriram Kosuri and George M Church. Large-scale de novo DNA synthesis: technologies and applications. *Nature methods*, 11(5):499–507, 2014. ISSN 1548-7105. doi: 10.1038/nmeth.2918.

Ryan A. LaCroix, Troy E. Sandberg, Edward J. O'Brien, Jose Utrilla, Ali Ebrahim, Gabriela I. Guzman, Richard Szubin, Bernhard O. Palsson, and Adam M. Feist. Use of adaptive laboratory evolution to discover key mutations enabling rapid growth of Escherichia coli K-12 MG1655 on glucose minimal medium. *Applied and Environmental Microbiology*, 81(1):17–30, 2015. ISSN 10985336. doi: 10.1128/AEM.02246-14.

Ryan A. LaCroix, Bernhard O. Palsson, and Adam M. Feist. A model for designing adaptive laboratory evolution experiments. *Applied and Environmental Microbiology*, 83(8), 2017. ISSN 10985336. doi: 10.1128/AEM.03115-16.

Warren Lau and Elizabeth S Sattely. Six enzymes from mayapple that complete the biosynthetic pathway to the etoposide aglycone. *Science*, 349(6253):1224–1228, 2015. ISSN 0036-8075. doi: 10.1126/science.aac7202.

Anna Lechner, Elizabeth Brunk, and Jay D Keasling. The Need for Integrated Approaches in Metabolic Engineering. *Cold Spring Harb Perspect Biol*, (August):a023903, 2016. ISSN 1943-0264. doi: 10.1101/cshperspect.a023903.

P C Lee and C Schmidt-Dannert. Metabolic engineering towards biotechnological production of carotenoids in microorganisms. *Applied microbiology and biotechnology*, 60:1–11, 2002. doi: 10.1007/s00253-002-1101-x.

Sang Yup Lee and Hyun Uk Kim. Systems strategies for developing industrial microbial strains. *Nature Biotechnology*, 33(10):1061–1072, 2015. ISSN 1087-0156. doi: 10.1038/nbt.3365.

John A. Lees and Stephen D. Bentley. Bacterial GWAS: not just gilding the lily. *Nature Reviews Microbiology*, 14(7):406–406, 2016. ISSN 1740-1526. doi: 10.1038/nrmicro.2016.82.

Qian Li, Zhiqiang Sun, Jing Li, and Yansheng Zhang. Enhancing beta-carotene production in *Saccharomyces cerevisiae* by metabolic engineering. *FEMS microbiology letters*, 345:94–101, 2013. doi: 10.1111/1574-6968.12187.

Peter E Lobban and A D Kaiser. Enzymatic end-to-end joining of DNA molecules. *Journal of Molecular Biology*, 78(3):453–471, 1973. ISSN 0022-2836. doi: 10.1016/0022-2836(73)90468-3.

Xiaoyan Ma and Yi-Xin Huo. The application of microfluidic-based technologies in the cycle of metabolic engineering. *Synthetic and Systems Biotechnology*, 1(3):137–142, sep 2016. ISSN 2405805X. doi: 10.1016/j.synbio.2016.09.004.

Daniel Machado and Markus J Herrgård. Co-evolution of strain design methods based on flux balance and elementary mode analysis. *Metabolic Engineering Communications*, 2:85–92, 2015. ISSN 2214-0301. doi: 10.1016/j.meteno.2015.04.001.

R. Mahadevan and C. H. Schilling. The effects of alternate optimal solutions in constraint-based genome-scale metabolic models. *Metabolic engineering*, 5(4):264–276, oct 2003. ISSN 10967176. doi: 10.1016/j.ymben.2003.09.002.

L.H. Mahishi, G. Tripathi, and S.K. Rawal. Poly(3-hydroxybutyrate) (PHB) synthesis by recombinant Escherichia coli harbouring Streptomyces aureofaciens PHB biosynthesis genes: Effect

of various carbon and nitrogen sources. *Microbiological Research*, 158(1):19–27, 2003. ISSN 09445013. doi: 10.1078/0944-5013-00161. URL http://linkinghub.elsevier.com/retrieve/pii/S0944501304700981.

Paulo Maia, Miguel Rocha, and Isabel Rocha. In Silico Constraint-Based Strain Optimization Methods: the Quest for Optimal Cell Factories. *Microbiology and molecular biology reviews : MMBR*, 80(1):45–67, 2016. ISSN 1098-5557. doi: 10.1128/MMBR.00014-15.

Mario A. Marchisio and Jörg Stelling. Computational design tools for synthetic biology. *Current Opinion in Biotechnology*, 20(4):479–485, 2009. ISSN 09581669. doi: 10.1016/j.copbio.2009.08.007.

Jan Marienhagen and Michael Bott. Metabolic engineering of microorganisms for the synthesis of plant natural products. *Journal of Biotechnology*, 163:166–178, 2013. ISSN 0168-1656. doi: 10.1016/j.jbiotec.2012.06.001.

Douglas McCloskey, Bernhard Ø Palsson, and Adam M Feist. Basic and applied uses of genome-scale metabolic network reconstructions of Escherichia coli. *Molecular Systems Biology*, 9(661): 661, jan 2013. ISSN 1744-4292. doi: 10.1038/msb.2013.18.

Janet E Mertz and Ronald W Davis. Cleavage of DNA by RI Restriction Endonuclease Generates Cohesive Ends. *Proc. Nat. Acad. Sci.*, 69(11):3370–3374, 1972. doi: 10.1073/pnas.69.11.3370.

J. F. Moxley, M. C. Jewett, M. R. Antoniewicz, S. G. Villas-Boas, H. Alper, R. T. Wheeler, L. Tong, A. G. Hinnebusch, T. Ideker, J. Nielsen, and G. Stephanopoulos. Linking high-resolution metabolic flux phenotypes and transcriptional regulation in yeast modulated by the global regulator Gcn4p. *Proceedings of the National Academy of Sciences*, 106(16):6477–6482, 2009. ISSN 0027-8424. doi: 10.1073/pnas.0811091106.

Kelly P. Nevin, Sarah A. Hensley, Ashley E. Franks, Zarath M. Summers, Jianhong Ou, Trevor L. Woodard, Oona L. Snoeyenbos-West, and Derek R. Lovley. Electrosynthesis of organic compounds from carbon dioxide is catalyzed by a diversity of acetogenic microorganisms. *Applied and Environmental Microbiology*, 77(9):2882–2886, 2011. ISSN 00992240. doi: 10.1128/AEM.02642-10.

Chiam Yu Ng, Moo-young Jung, Jinwon Lee, and Min-Kyu Oh. Production of 2,3-butanediol in Saccharomyces cerevisiae by in silico aided metabolic engineering. *Microbial Cell Factories*, 11(1):68, 2012. ISSN 1475-2859. doi: 10.1186/1475-2859-11-68.

Jens Nielsen and Jay D Keasling. Engineering Cellular Metabolism. *Cell*, 164(6):1185–1197, 2016. ISSN 0092-8674. doi: 10.1016/j.cell.2016.02.004.

Edward J. O'Brien and Bernhard O Palsson. Computing the functional proteome: recent progress and future prospects for genome-scale models. *Current Opinion in Biotechnology*, 34:125–134, 2015. ISSN 0958-1669. doi: 10.1016/j.copbio.2014.12.017.

Edward J. O'Brien, Jonathan M. Monk, and Bernhard O. Palsson. Using genome-scale models to predict biological capabilities. *Cell*, 161(5):971–987, 2015. ISSN 10974172. doi: 10.1016/j.cell.2015.05.019.

Jeffrey D. Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–8, mar 2010. ISSN 1546-1696. doi: 10.1038/nbt.1614.

Jeffrey D Orth, Tom M Conrad, Jessica Na, Joshua A Lerman, Hojung Nam, Adam M Feist, and Bernhard Ø Palsson. A comprehensive genome-scale reconstruction of *Escherichia coli* metabolism—2011. *Molecular Systems Biology*, 7:535–535, 2011. ISSN 1744-4292. doi: 10.1038/msb.2011.65.

Kiran Raosaheb Patil, Isabel Rocha, Jochen Förster, and Jens Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC bioinformatics*, 6:308, jan 2005. ISSN 1471-2105. doi: 10.1186/1471-2105-6-308.

Pamela P. Peralta-Yahya, Fuzhong Zhang, Stephen B. del Cardayre, and Jay D. Keasling. Microbial engineering for the production of advanced biofuels. *Nature*, 488(7411):320–328, 2012. ISSN 0028-0836. doi: 10.1038/nature11478.

Christopher J. Petzold, G. Leanne Jade Chan, Melissa Nhan, and Paul D. Adams. Analytics for metabolic engineering. *Frontiers in Bioengineering and Biotechnology*, 3(September):1–11, 2015. doi: 10.3389/fbioe.2015.00135.

Priti Pharkya, Anthony P Burgard, and Costas D Maranas. OptStrain: a computational framework for redesign of microbial production systems. *Genome research*, 14(11):2367–76, nov 2004. ISSN 1088-9051. doi: 10.1101/gr.2872004.

Celeste C. Quianzon and Issam Cheikh. History of insulin. *Journal of Community Hospital Internal Medicine Perspectives*, 2(2):18701, 2012. doi: 10.3402/jchimp.v2i2.18701.

Isabel Rocha, Paulo Maia, Pedro Evangelista, Paulo Vilaça, Simão Soares, José P Pinto, Jens Nielsen, Kiran R Patil, Eugénio C Ferreira, and Miguel Rocha. OptFlux: an open-source software platform for in silico metabolic engineering. *BMC Systems Biology*, 4(1):45, jan 2010. ISSN 1752-0509. doi: 10.1186/1752-0509-4-45.

Noah A. Rosenberg, Lucy Huang, Ethan M. Jewett, Zachary A. Szpiech, Ivana Jankovic, and Michael Boehnke. Genome-wide association studies in diverse populations. *Nature Reviews Genetics*, 11(5):356–366, 2010. ISSN 1471-0056. doi: 10.1038/nrg2760.

Douglas T. Ross. Computer-aided design: A statement of objectives. Technical Report 8436-TM-4, MIT Servo Lab., Cambridge, Massachusetts, 1960.

Harmen M Van Rossum, Barbara U Kozak, Jack T Pronk, and Antonius J A Van Maris. Engineering cytosolic acetyl-coenzyme A supply in *Saccharomyces cerevisiae*: Pathway stoichiometry, free-energy conservation and redox-cofactor balancing. *Metabolic Engineering*, 36:99–115, 2016a. ISSN 1096-7176. doi: 10.1016/j.ymben.2016.03.006.

Harmen M Van Rossum, Barbara U. Kozak, Jack T. Pronk, Antonius J A Van Maris, Harmen M. van Rossum, Barbara U. Kozak, Jack T. Pronk, and Antonius J.A. van Maris. Engineering cytosolic acetyl-coenzyme A supply in Saccharomyces cerevisiae: Pathway stoichiometry, free-energy conservation and redox-cofactor balancing. *Metabolic Engineering*, 36:99–115, jul 2016b. ISSN 10967176. doi: 10.1016/j.ymben.2016.03.006.

R T Rowlands. Industrial strain improvement: rational screens and genetic recombination techniques. *Enzyme and Microbial Technology*, 6(1):3–10, 1984. doi: 10.1016/0141-0229(84)90070-X.

Handunkutti P.V Rupasinghe, Kurt C Almquist, Gopinadhan Paliyath, and Dennis P Murr. Cloning of hmg1 and hmg2 cDNAs encoding 3-hydroxy-3-methylglutaryl coenzyme A reductase and their expression and activity in relation to α-farnesene synthesis in apple. *Plant Physiology and Biochemistry*, 39(11):933–947, nov 2001. ISSN 09819428. doi: 10.1016/S0981-9428(01)01316-X.

Benjamín J Sánchez, Cheng Zhang, Avlant Nilsson, Petri-Jaan Lahtvee, Eduard J. Kerkhoven, and Jens Nielsen. Improving the phenotype predictions of a yeast genome-scale metabolic model by incorporating enzymatic constraints. *Molecular Systems Biology*, 13(8):935, aug 2017. ISSN 1744-4292. doi: 10.15252/msb.20167411.

Jeffry D Sander and J Keith Joung. CRISPR-Cas systems for editing, regulating and targeting genomes. *Nature biotechnology*, 32(4):347–55, 2014. ISSN 1546-1696. doi: 10.1038/nbt.2842.

Anand Sastry, Jonathan Monk, Hanna Tegel, Mathias Uhlen, Bernhard O. Palsson, Johan Rockberg, and Elizabeth Brunk. Machine learning in computational biology to accelerate high-throughput protein expression. *Bioinformatics*, apr 2017. ISSN 1367-4803. doi: 10.1093/bioinformatics/btx207.

U Sauer. Evolutionary engineering of industrially important microbial phenotypes. *Advances in biochemical engineering/biotechnology*, 73:129–169, 2001. ISSN 0724-6145. doi: 10.1007/3-540-45300-8_7.

Brian J. Schmidt, Ali Ebrahim, Thomas O. Metz, Joshua N. Adkins, Daniel R Palsson Bernhard Øand Hyduke, Bernhard Ø Palsson, and Daniel R. Hyduke. GIM3E: condition-specific models of cellular metabolism developed from metabolomics and expression data. *Bioinformatics (Oxford, England)*, 29(22):2900–2908, 2013. ISSN 1367-4811. doi: 10.1093/bioinformatics/btt493.

Daniel Segrè, Dennis Vitkup, and George M Church. Analysis of optimality in natural and perturbed metabolic networks. *Proceedings of the National Academy of Sciences of the United States of America*, 99(23):15112–7, nov 2002. ISSN 0027-8424. doi: 10.1073/pnas.232349399.

Tomer Shlomi, Omer Berkman, and Eytan Ruppin. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *Proceedings of the National Academy of Sciences of the United States of America*, 102(21):7695–700, may 2005. ISSN 0027-8424. doi: 10.1073/pnas.0406346102.

Michael J Smanski, Swapnil Bhatia, Dehua Zhao, Yongjin Park, Lauren B A Woodruff, Georgia Giannoukos, Dawn Ciulla, Michele Busby, Johnathan Calderon, Robert Nicol, D Benjamin Gordon, Douglas Densmore, and Christopher A Voigt. Functional optimization of gene clusters by combinatorial design and assembly. *Nature Biotechnology*, 32(12), 2014. ISSN 1087-0156. doi: 10.1038/nbt.3063.

Kim I. Sørensen, Mirjana Curic-Bawden, Mette P. Junge, Thomas Janzen, and Eric Johansen. Enhancing the sweetness of yoghurt through metabolic remodeling of carbohydrate metabolism in Streptococcus thermophilus and Lactobacillus delbrueckii subsp. bulgaricus. *Applied and Environmental Microbiology*, 82(April):AEM.00462–16, 2016. ISSN 0099-2240. doi: 10.1128/AEM.00462-16.

Jan Steensels, Tim Snoek, Esther Meersman, Martina Picca Nicolino, Karin Voordeckers, and Kevin J. Verstrepen. Improving industrial yeast strains: Exploiting natural and artificial diversity. *FEMS Microbiology Reviews*, 38(5):947–995, 2014. ISSN 15746976. doi: 10.1111/1574-6976.12073.

Naama Tepper and Tomer Shlomi. Predicting metabolic engineering knockout strategies for chemical production: accounting for competing pathways. *Bioinformatics (Oxford, England)*, 26(4):536–43, feb 2010. ISSN 1367-4811. doi: 10.1093/bioinformatics/btp704.

Don Tribby. *Yogurt*, pages 191–223. Springer US, New York, NY, 2009. ISBN 978-0-387-77408-4. doi: 10.1007/978-0-387-77408-4_8.

United Nations. Convention on Biological Diversity. In *Treaty Series*. United Nations, 2001. URL https://treaties.un.org/doc/Publication/UNTS/Volume{%}201760/v1760.pdf.

Axel von Kamp and Steffen Klamt. Enumeration of smallest intervention strategies in genome-scale metabolic networks. *PLoS computational biology*, 10(1):e1003378, jan 2014. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1003378.

Tilmann Weber, Pep Charusanti, Ewa Maria Musiol-Kroll, Xinglin Jiang, Yaojun Tong, Hyun Uk Kim, and Sang Yup Lee. Metabolic engineering of antibiotic factories: new tools for antibiotic production in actinomycetes. *Trends in Biotechnology*, 33(1):15–26, 2015. ISSN 0167-7799. doi: 10.1016/j.tibtech.2014.10.009.

Peng Xu, Elizabeth Anne Rizzoni, Se Yeong Sul, and Gregory Stephanopoulos. Improving Metabolic Pathway Efficiency by Statistical Model-Based Multivariate Regulatory Metabolic Engineering. *ACS synthetic biology*, 6(1):148–158, 2017. ISSN 21615063. doi: 10.1021/acssynbio.6b00187.

Jung Eun Yang, Je Woong Kim, Young Hoon Oh, So Young Choi, Hyuk Lee, A-Reum Park, Jihoon Shin, Si Jae Park, and Sang Yup Lee. Biosynthesis of poly(2-hydroxyisovalerate-co-lactate) by metabolically engineered Escherichia coli. *Biotechnology Journal*, 11(12):1572–1585, dec 2016. ISSN 18606768. doi: 10.1002/biot.201600420.

K. Yizhak, T. Benyamini, W. Liebermeister, E. Ruppin, and Tomer Shlomi. Integrating quantitative proteomics and metabolomics with a genome-scale metabolic network model. *Bioinformatics*, 26(12):i255–i260, jun 2010. ISSN 1367-4803. doi: 10.1093/bioinformatics/btq183.

Thomas W. Young. Beer. Online, 2017. URL https://www.britannica.com/topic/beer{%}0A.

Jürgen Zanghellini, David E. Ruckerbauer, Michael Hanscho, and Christian Jungreuthmayer. Elementary flux modes in a nutshell: Properties, calculation and applications. *Biotechnology Journal*, 8(9):1009–1016, sep 2013. ISSN 18606768. doi: 10.1002/biot.201200269.

X Zhang, K Jantama, K T Shanmugam, and L O Ingram. Reengineering *Escherichia coli* for Succinate Production in Mineral Salts Medium. *Applied and Environmental Microbiology*, 75(24):7807–7813, 2009. doi: 10.1128/AEM.01758-09.

Shengde Zhou, T B Causey, A Hasona, K T Shanmugam, and L O Ingram. Production of Optically Pure D-Lactic Acid in Mineral Salts Medium by Metabolically Engineered *Escherichia coli* W3110 †. *Applied and Environmental Microbiology*, 69(1):399–407, 2003. doi: 10.1128/AEM.69.1.399.

Hadas Zur, Eytan Ruppin, and Tomer Shlomi. iMAT: an integrative metabolic analysis tool. *Bioinformatics (Oxford, England)*, 26(24):3140–2, dec 2010. ISSN 1367-4811. doi: 10.1093/bioinformatics/btq602.

It is all about metabolic fluxes.

Jens Nielsen

2

Cameo: A Python Library for Computer Aided Metabolic Engineering and Optimization of Cell Factories

Summary

This chapter describes *cameo*, a CAD software for cell factory design. This tool is easy to use and provides novel and state-of-the-art algorithms to identify the genetic modifications that can be used to improve cell factories. This work has been deposited on BioRxiv on 9 June 2017 (https://doi.org/10.1101/147199).

Abstract

Computational systems biology methods enable rational design of cell factories on a genome-scale and thus accelerate the engineering of cells for the production of valuable chemicals and proteins. Unfortunately, for the majority of these methods' implementations are either not published, rely on proprietary software, or do not provide documented interfaces, which has precluded their mainstream adoption in the field. In this work we present cameo, a platform-independent software that enables *in silico* design of cell factories and targets both experienced modelers as well as users new to the field. It is written in Python and implements state-of-the-art methods for enumerating and prioritizing knock-out, knock-in, over-expression, and down-regulation strategies and combinations thereof. Cameo is an open source software project and is freely available under the Apache License 2.0. A dedicated website including documentation, examples, and installation instructions can be found at `http://cameo.bio`. Users can also give cameo a try at `http://try.cameo.bio`.

Introduction

The engineering of cells for the production of chemicals and proteins affects all areas of our modern lives. Beer, yogurt, flavoring, detergents, and insulin represent just a few products which are unimaginable without biotechnology. Engineered cells may further provide solutions to many of mankind's greatest challenges like global climate, multiple drug resistance, and overpopulation, by producing fuels, novel antibiotics, and food from renewable feedstocks. Manipulating cells to perform tasks that they did not evolved for, however, is challenging and requires significant investments and personnel in order to reach economically viable production of target molecules (Lee and Kim, 2015).

A central task in developing biotechnological production processes is to reroute metabolic fluxes towards desired products in cells. This task is particularly prone to failure due to our limited understanding of the underlying biology and the complexity of the metabolic networks in even the simplest of organisms. In line with other recent technological advancements, like high-fidelity genome editing through CRISPR/Cas9 (Sander and Joung, 2014) and DNA synthesis costs dropping (Kosuri and Church, 2014), modeling methods are increasingly used to accelerate

cell factory engineering, helping to reduce development time and cost (Meadows et al., 2016).

Genome-scale models of metabolism (GEMs) (McCloskey et al., 2013) are of particular interest in this context as they predict phenotypic consequences of genetic and environmental perturbations affecting cellular metabolism (O'Brien et al., 2015). These models have been developed throughout the past 15 years for the majority of potential cell factory host organisms ranging from bacteria to mammalian cells. A large repertoire of algorithms has been published that utilize GEMs to compute cell factory engineering strategies composed of over-expression, down-regulation, deletion, and addition of genes (see Machado and Herrgård, 2015, Maia et al., 2016). Unfortunately, most of these algorithms are not easily accessible to users as they have either been published without implementation (e.g. using pseudo code or mathematical equations to describe the method) or the implementation provided by the authors is undocumented or hard to install. These problems significantly limit the ability of metabolic engineers to utilize computational design tools as part of their workflow.

Results

Cameo is open source software written in Python that alleviates these problems and aims to make *in silico* cell factory design broadly accessible. On the one hand it enables cell factory engineers to enumerate and prioritize designs without having to be experts in metabolic modeling themselves. On the other hand it aims to become a comprehensive library of published methods by providing method developers with a library that simplifies the implementation of new cell factory design methods.

Cameo provides a high-level interface that can be used without knowing any metabolic modeling or how different algorithms are implemented (see Supplementary Notebook 8 [v0.10.3, current]). In fact, the most minimal form of input that cameo requires is simply the desired product, for example vanillin.

```
from cameo import api
api.design(product='vanillin')
```

This function call will run the workflow depicted in Figure 2.1. It is also possible to call the same functionality from the command line. Firstly, it enumerates native and heterologous pro-

duction pathways for a series of commonly used host organisms and carbon sources. Then it runs a whole suite of design algorithms available in cameo to generate a list of metabolic engineering strategies, which can then be ranked by different criteria (maximum theoretical yield, number of genetic modifications etc.).

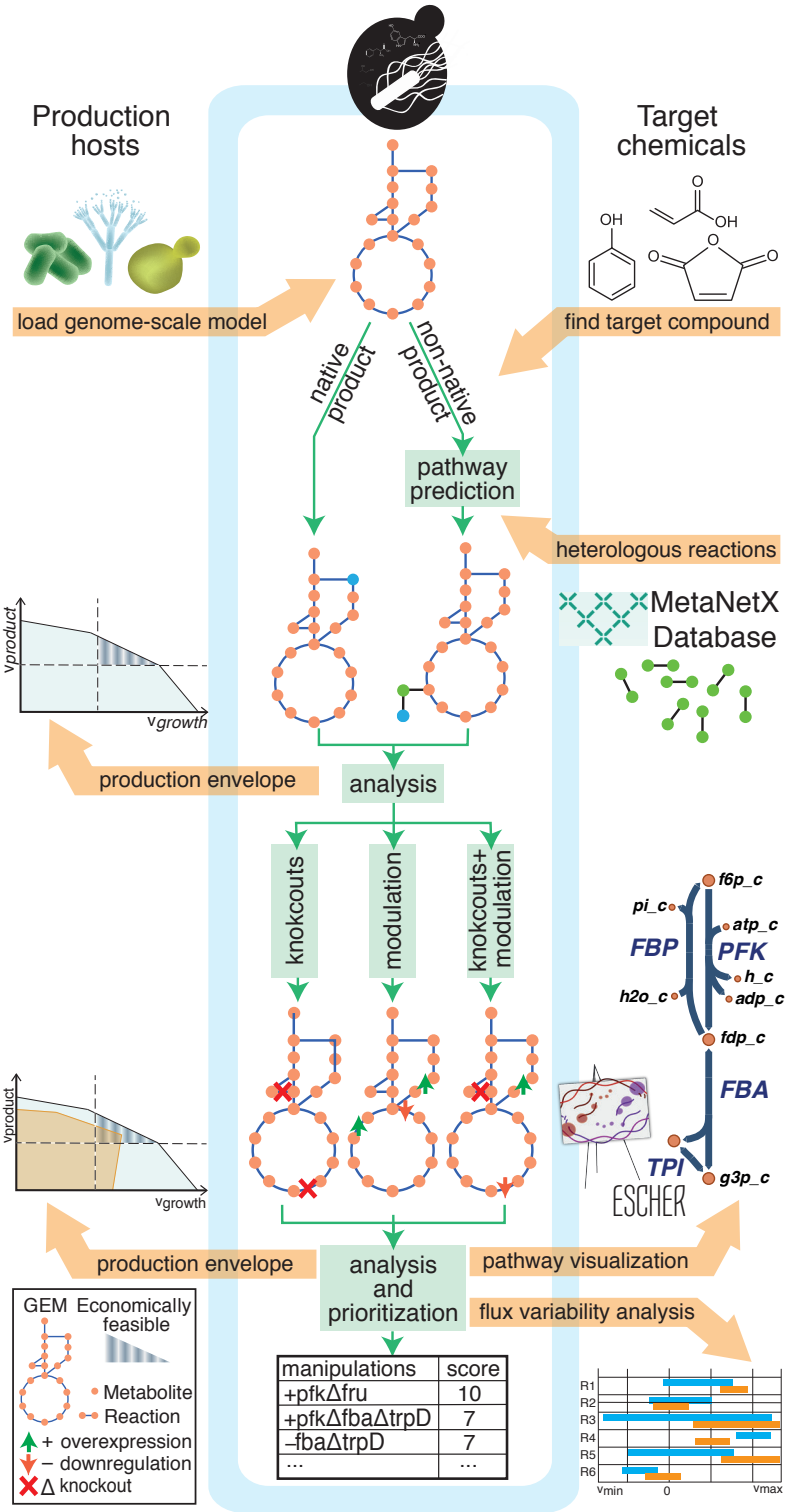
More advanced users can easily customize this workflow by providing models for other host organisms, changing parameters and algorithms, and of course by including their own methods.

In order to become a community project and attract further developers, cameo has been developed as a modular Python package that has been extensively documented and tested using modern software engineering practices like test-driven development and continuous integration/deployment on travis-ci.org (Figure 2.2 shows an overview of the package organization).

To avoid duplication of effort, cameo is based on the constraint-based modeling tool cobrapy (Ebrahim et al., 2013) thus providing its users with already familiar objects and methods (see also Figure 2.2a). Furthermore, cameo takes advantage of other popular tools of the scientific Python stack, like for example Jupyter notebooks for providing an interactive modeling environment (Pérez and Granger, 2007) and pandas for the representation, querying, and visualization of results (McKinney, 2010).

Accessing published GEMs can be a challenging task as they are often made available in formats that are not supported by existing modeling software (Ebrahim et al., 2015). Cameo provides programmatic access to collections of models (Figure 2.2b) hosted by BiGG (King et al., 2016) and the University of Minho darwin.di.uminho.pt/models. Furthermore, by relying on the common namespace for reaction and metabolite identifiers provided by the MetaNetX.org project (Bernard et al., 2014) that covers commonly used pathway databases like KEGG (Kanehisa et al., 2016), RHEA (Morgat et al., 2015), and BRENDA (Chang et al., 2015), a universal reaction

Figure 2.1 ***(following page)*****: Cell factory design workflow with cameo.** The first step is to import a metabolic model from a file or using a web service. Next, the user needs to select a target product. If the target product is a non-native chemical, shortest heterologous production pathways can be enumerated to determine a suitable route to the product (Pharkya et al., 2004). Potential production pathways can then be compared using production envelopes, i.e., visualizations of the trade-off between production rate and organism growth rate (see Supplementary Notebook 4 [v0.10.3, current]). After a production pathway has been chosen, a number of different design methods are used to compute the genetic modifications (designs) necessary to achieve the production goal (see Supplementary Notebooks 5 [v0.10.3, current] and 6 [v0.10.3, current]). In the end, the computed designs can be sorted using different criteria relevant to the actual implementation in the lab and economic considerations such as the number of genetic modifications needed and maximum theoretical product yield. Furthermore, a number of results can be further visualized using the pathway visualization tool Escher (King et al., 2015)


Production hosts
Target chemicals
load genome-scale model
find target compound
native product
non-native product
pathway prediction
heterologous reactions
MetaNetX Database
production envelope
analysis
knockouts
modulation
knockouts+ modulation
f6p_c
pi_c
atp_c
FBP
PFK
h_c
h2o_c
adp_c
fdp_c
FBA
TPI
g3p_c
ESCHER
production envelope
analysis and prioritization
pathway visualization
flux variability analysis
GEM
Economically feasible
Metabolite
Reaction
+ overexpression
– downregulation
Δ knockout
manipulations | score
+pfkΔfru | 10
+pfkΔfbaΔtrpD | 7
-fbaΔtrpD | 7
... | ...
R1
R2
R3
R4
R5
R6
vmin
0
vmax

database can be used to predict heterologous pathways (see Supplementary Notebook 7 [v0.10.3, current].

Most design algorithms rely on solving optimization problems. In order to speed up simulations and ease the formulation of optimization problems, cameo replaces the solver interfaces utilized in cobrapy with optlang (Jensen et al., 2017), a Python interface to commonly used optimization solvers and symbolic modeling language that is maintained by the authors of cameo. Cameo always maintains a one-to-one correspondence of the GEM and its underlying optimization problem, greatly facilitating debugging and efficient solving by enabling warm starts from previously found solutions (Gelius-Dietrich et al., 2013). Furthermore, being based on sympy (SymPy Development Team, 2016), optlang enables the formulation of complicated optimization problems using symbolic math expressions, making the implementation of published design methods straightforward.

Runtimes of design methods are usually on the order of seconds to minutes. Nevertheless, scanning large numbers of potential products, host organisms, and feedstocks, can quickly make computations challenging (running the entire workflow using the high-level API takes on the order of hours). As described above, cameo makes unit operations as fast as possible by implementing an efficient interface to the underlying optimization software. In addition, a number of methods in cameo can be parallelized, and can thus take advantage of multicore CPUs and HPC infrastructure if available (see documentation).

With this broad overview of capabilities, we would like to emphasize the role of cameo as a useful resource to the modeling community and wish to support its development as a community effort in the long run. The majority of published strain design algorithms have not been experimentally validated (Machado and Herrgård, 2015) and we believe that their inaccessibility to users is a major factor for the lack of validation. With cameo we hope to counteract this problem by making these methods accessible to the entire metabolic engineering community and also providing a platform for modelers to implement and publish novel methods.

Conclusions

With cameo version 0.10.3 we release a tool that is ready to be used in metabolic engineering projects. It is under active development and future work will include interfacing cameo with

genome-editing tools to streamline the translation of computed strain designs into laboratory protocols, modeling of fermentation processes to get estimates on titers and productivities, and include pathway predictions based on retrobiosynthesis including hypothetical biochemical conversions (Campodonico et al., 2014).

Acknowledgments

We would like to thank Kai Zhuang, Miguel Campodonico and Sumesh Sukumura for providing valuable feedback and bug reports as early users of cameo. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 686070. Furthermore, we acknowledge financial support from the Novo Nordisk Foundation.

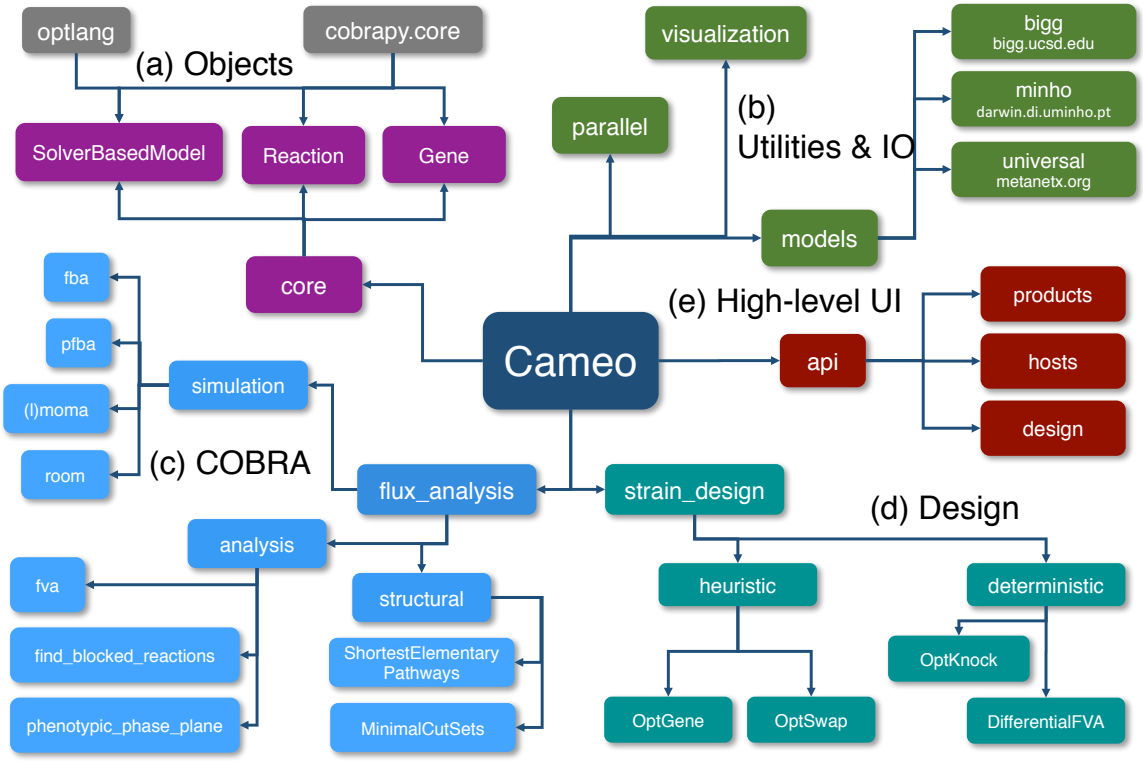


Figure 2.2: Package organization and functionality overview. The cameo package is organized into a number of sub-packages: *core* extends cobrapy's own *core* package (Ebrahim et al., 2015) to use optlang (Jensen et al., 2017) as interface to a number of optimization solvers. *models* enables programmatic access to models hosted on the internet. *parallel* provides tools for the parallelization of design methods. *visualization* provides a number of high-level visualization functions, e.g., production envelopes. *flux_analysis* implements many basic simulation and analysis methods needed for higher-level design methods and the evaluation of production goals etc. *strain_design* provides a collection of *in silico* design methods and is subdivided into methods that use deterministic and heuristic optimization approaches. At last, *api* provides a high-level interface for computing designs.

References

Thomas Bernard, Alan Bridge, Anne Morgat, Sébastien Moretti, Ioannis Xenarios, and Marco Pagni. Reconciliation of metabolites and biochemical reactions for metabolic networks. *Briefings in Bioinformatics*, 15(1):123–135, 2014. ISSN 14675463. doi: 10.1093/bib/bbs058.

Miguel A. Campodonico, Barbara A. Andrews, Juan A. Asenjo, Bernhard O. Palsson, and Adam M. Feist. Generation of an atlas for commodity chemical production in *Escherichia coli* and a novel pathway prediction algorithm, GEM-Path. *Metabolic Engineering*, 25:140–158, 2014. ISSN 10967184. doi: 10.1016/j.ymben.2014.07.009.

Antje Chang, Ida Schomburg, Sandra Placzek, Lisa Jeske, Marcus Ulbrich, Mei Xiao, Christoph W. Sensen, and Dietmar Schomburg. BRENDA in 2015: Exciting developments in its 25th year of existence. *Nucleic Acids Research*, 43(D1):D439–D446, 2015. ISSN 13624962. doi: 10.1093/nar/gku1068.

A Ebrahim, JA Lerman, BO Palsson, and DR Hyduke. COBRApy: COnstraints-Based Reconstruction and Analysis for Python. *BMC Syst Biol*, 7:74, Aug 2013.

Ali Ebrahim, Eivind Almaas, Eugen Bauer, Aarash Bordbar, Anthony P Burgard, Roger L Chang, A. Dräger, Iman Famili, Adam M Feist, R. M. Fleming, Stephen S Fong, Vassily Hatzimanikatis, Markus J Herrgard, Allen Holder, Michael Hucka, Daniel Hyduke, Neema Jamshidi, Sang Yup Lee, N. Le Novere, Joshua A Lerman, Nathan E Lewis, Ding Ma, Radhakrishnan Mahadevan, Costas Maranas, Harish Nagarajan, Ali Navid, J. Nielsen, Lars K Nielsen, Juan Nogales, Alberto Noronha, C. Pal, Bernhard O. Palsson, Jason A Papin, Kiran R Patil, Nathan D Price, Jennifer L Reed, Michael Saunders, Ryan S Senger, Nikolaus Sonnenschein, Yuekai Sun, and Ines Thiele. Do genome-scale models need exact solvers or clearer standards? *Molecular Systems Biology*, 11(10):831–831, oct 2015. ISSN 1744-4292. doi: 10.15252/msb.20156157.

Gabriel Gelius-Dietrich, Abdelmoneim Amer Desouki, Claus Jonathan Fritzemeier, and Martin J Lercher. Sybil–efficient constraint-based modelling in R. *BMC systems biology*, 7:125, 2013. ISSN 1752-0509. doi: 10.1186/1752-0509-7-125.

Kristian Jensen, Joao G.R. Cardoso, and Nikolaus Sonnenschein. Optlang: An algebraic modeling language for mathematical optimization. *The Journal of Open Source Software*, 2(9), jan 2017. doi: 10.21105/joss.00139. URL https://doi.org/10.21105%2Fjoss.00139.

Minoru Kanehisa, Yoko Sato, Masayuki Kawashima, Miho Furumichi, and Mao Tanabe. KEGG as a reference resource for gene and protein annotation. *Nucleic Acids Research*, 44(D1):D457–D462, 2016. ISSN 13624962. doi: 10.1093/nar/gkv1070.

Zachary A. King, Andreas Dräger, Ali Ebrahim, Nikolaus Sonnenschein, Nathan E. Lewis, and Bernhard O. Palsson. Escher: A Web Application for Building, Sharing, and Embedding Data-Rich Visualizations of Biological Pathways. *PLOS Computational Biology*, 11(8):e1004321, aug 2015. ISSN 1553-7358. doi: 10.1371/journal.pcbi.1004321.

Zachary A. King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A. Lerman, Ali Ebrahim, Bernhard O. Palsson, and Nathan E. Lewis. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44(D1): D515–D522, jan 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1049.

Sriram Kosuri and George M Church. Large-scale de novo DNA synthesis: technologies and applications. *Nature methods*, 11(5):499–507, 2014. ISSN 1548-7105. doi: 10.1038/nmeth.2918.

Sang Yup Lee and Hyun Uk Kim. Systems strategies for developing industrial microbial strains. *Nat Biotech*, 33(10):1061–1072, 10 2015. doi: 10.1038/nbt.3365.

Daniel Machado and Markus J. Herrgård. Co-evolution of strain design methods based on flux balance and elementary mode analysis. *Metabolic Engineering Communications*, 2:85 – 92, 2015. ISSN 2214-0301. doi: j.meteno.2015.04.001.

Paulo Maia, Miguel Rocha, and Isabel Rocha. *In Silico* Constraint-Based Strain Optimization Methods: the Quest for Optimal Cell Factories. *Microbiology and molecular biology reviews*, 80(1):45–67, 2016. ISSN 1098-5557. doi: 10.1128/MMBR.00014-15.

Douglas McCloskey, Bernhard ØPalsson, and Adam M Feist. Basic and applied uses of genome-scale metabolic network reconstructions of *Escherichia coli*. *Molecular Systems Biology*, 9(661): 661, jan 2013. ISSN 1744-4292. doi: 10.1038/msb.2013.18.

Wes McKinney. Data Structures for Statistical Computing in Python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

Adam L. Meadows, Kristy M. Hawkins, Yoseph Tsegaye, Eugene Antipov, Youngnyun Kim, Lauren Raetz, Robert H. Dahl, Anna Tai, Tina Mahatdejkul-Meadows, Lan Xu, Lishan Zhao, Madhukar S. Dasika, Abhishek Murarka, Jacob Lenihan, Diana Eng, Joshua S. Leng, Chi-Li Liu, Jared W. Wenger, Hanxiao Jiang, Lily Chao, Patrick Westfall, Jefferson Lai, Savita Ganesan, Peter Jackson, Robert Mans, Darren Platt, Christopher D. Reeves, Poonam R. Saija, Gale Wichmann, Victor F. Holmes, Kirsten Benjamin, Paul W. Hill, Timothy S. Gardner, and Annie E. Tsong. Rewriting yeast central carbon metabolism for industrial isoprenoid production. *Nature*, pages 1–16, 2016. ISSN 0028-0836. doi: 10.1038/nature19769.

Anne Morgat, Kristian B. Axelsen, Thierry Lombardot, Rafael Alcántara, Lucila Aimo, Mohamed Zerara, Anne Niknejad, Eugeni Belda, Nevila Hyka-Nouspikel, Elisabeth Coudert, Nicole Redaschi, Lydie Bougueleret, Christoph Steinbeck, Ioannis Xenarios, and Alan Bridge. Updates in Rhea-a manually curated resource of biochemical reactions. *Nucleic Acids Research*, 43(D1):D459–D464, 2015. ISSN 13624962. doi: 10.1093/nar/gku961.

Edward J. O'Brien, Jonathan M. Monk, and Bernhard O. Palsson. Using genome-scale models to predict biological capabilities. *Cell*, 161(5):971–987, 2015. ISSN 10974172. doi: 10.1016/j.cell.2015.05.019.

Fernando Pérez and Brian E. Granger. IPython: a system for interactive scientific computing. *Computing in Science and Engineering*, 9(3):21–29, May 2007. ISSN 1521-9615. doi: 10.1109/MCSE.2007.53.

Priti Pharkya, Anthony P. Burgard, and Costas D. Maranas. OptStrain: A computational framework for redesign of microbial production systems. *Genome Research*, 14(11):2367–2376, 2004. ISSN 10889051. doi: 10.1101/gr.2872004.

Jeffry D Sander and J Keith Joung. CRISPR-Cas systems for editing, regulating and targeting genomes. *Nature biotechnology*, 32(4):347–55, 2014. ISSN 1546-1696. doi: 10.1038/nbt.2842.

SymPy Development Team. *SymPy: Python library for symbolic mathematics*, 2016. URL http://www.sympy.org.

Supplementary Notebooks

The live version of these notebooks can be found online at: http://try.cameo.bio/

01-quick-start

August 13, 2017

1 Getting started with cameo

cameo reuses and extends model data structures defined by cobrapy (**CO**nstraints-**B**ased **R**econstruction and **A**nalysis tool for **Py**thon). So, in addition to following this quick start guide and other **cameo** tutorials, we encourage you to explore cobrapy's documentation as well.

1.1 Step 1: Load a model

Loading a model is easy. Just import the `~cameo.io.load_model` function.

```
In [1]: from cameo import load_model
```

For example, load the most current genome-scale metabolic reconstruction of *Escherichia coli*.

```
In [2]: model = load_model("iJO1366")
```

Models, reactions, metabolites, etc., provide return HTML when evaluated in Jupyter notebooks and can thus be easily inspected.

```
In [3]: model
```

```
Out[3]: <Model iJO1366 at 0x115c8ceb8>
```

1.2 Step 2: Simulate a model

The model can be simulated by executing `~cameo.core.solver_based_model.SolverBasedModel.solve`.

```
In [4]: solution = model.optimize()
```

A quick overview of the solution can be obtained in form of a pandas DataFrame (all solution objects in cameo provide access to data frames through a `data_frame` attribute).

```
In [5]: solution
```

```
Out[5]: <Solution 0.982 at 0x11555b160>
```

A data frame representation of the solution is accessible via `solution.to_frame()`.

```
In [6]: solution.to_frame()
```

```
Out[6]:                  fluxes  reduced_costs
        DM_4crsol_c  2.1907e-04         0.0000
        DM_5drib_c   2.2103e-04         0.0000
        DM_aacald_c -0.0000e+00         0.0000
        DM_amob_c    1.9647e-06         0.0000
        DM_mththf_c  4.4010e-04         0.0000
        ...                 ...            ...
        ZN2abcpp     0.0000e+00        -0.0083
        ZN2t3pp      0.0000e+00        -0.0021
        ZN2tpp       3.3499e-04         0.0000
        ZNabcpp      0.0000e+00        -0.0083
        Zn2tex       3.3499e-04        -0.0000

        [2583 rows x 2 columns]
```

Data frames make it very easy to process results. For example, let's take a look at reactions with flux != 0

```
In [7]: solution.to_frame().query('fluxes != 0')
```

```
Out[7]:                                      fluxes  reduced_costs
        DM_4crsol_c                     2.1907e-04     0.0000e+00
        DM_5drib_c                      2.2103e-04     0.0000e+00
        DM_amob_c                       1.9647e-06     0.0000e+00
        DM_mththf_c                     4.4010e-04     0.0000e+00
        BIOMASS_Ec_iJO1366_core_53p95M  9.8237e-01     1.8492e-15
        ...                                    ...            ...
        UPPDC1                          2.1907e-04     0.0000e+00
        USHD                            1.9113e-02     0.0000e+00
        VALTA                          -4.1570e-01     0.0000e+00
        ZN2tpp                          3.3499e-04     0.0000e+00
        Zn2tex                          3.3499e-04    -0.0000e+00

        [437 rows x 2 columns]
```

1.3 Step 3: Exploring a model

Objects—models, reactions, metabolites, genes—can easily be explored in the Jupyter notebook, taking advantage of tab completion. For example, place your cursor after the period in `model.reactions.` and press the TAB key. A dialog will appear that allows you to navigate the list of reactions encoded in the model.

```
In [8]: model.reactions.PGK # delete PGK, place your cursor after the period and press the TAB
```

```
Out[8]: <Reaction PGK at 0x11643af98>
```

For example, you can access the E4PD (*Erythrose 4-phosphate dehydrogenase*) reaction in the model.

```
In [9]: model.reactions.E4PD
```

```
Out[9]: <Reaction E4PD at 0x116162c18>
```

Be aware though that due variable naming restrictions in Python dot notation access to reactions (and other objects) might not work in some cases.

```
In [10]: # model.reactions.12DGR120tipp  # uncommenting and running this cell will produce a s
```

In these cases you need to use the `model.reactions.get_by_id`.

```
In [11]: model.reactions.get_by_id('12DGR120tipp')
```

```
Out[11]: <Reaction 12DGR120tipp at 0x115f85d30>
```

Metabolites are accessible through `model.metabolites`. For example, D-glucose in the cytosolic compartment.

```
In [12]: model.metabolites.glc__D_c
```

```
Out[12]: <Metabolite glc__D_c at 0x115cfb898>
```

And it is easy to find the associated reactions

```
In [13]: model.metabolites.glc__D_c.reactions
```

```
Out[13]: frozenset({<Reaction MLTG1 at 0x1163779b0>,
                    <Reaction MLTG2 at 0x1163779e8>,
                    <Reaction TRE6PH at 0x116557ac8>,
                    <Reaction G6PP at 0x116206b00>,
                    <Reaction MLTG3 at 0x116377ba8>,
                    <Reaction GLCabcpp at 0x11623f3c8>,
                    <Reaction MLTG4 at 0x116377c18>,
                    <Reaction AMALT2 at 0x11605b438>,
                    <Reaction GLCt2pp at 0x11623f470>,
                    <Reaction MLTG5 at 0x116377c88>,
                    <Reaction TREH at 0x116557d30>,
                    <Reaction AMALT1 at 0x11605b588>,
                    <Reaction AMALT3 at 0x11605b6a0>,
                    <Reaction GLCATr at 0x1162336a0>,
                    <Reaction AMALT4 at 0x11605b710>,
                    <Reaction XYLI2 at 0x1165a1f28>,
                    <Reaction HEX1 at 0x1162a7748>,
                    <Reaction LACZ at 0x1162f0f98>,
                    <Reaction GALS3 at 0x1162157f0>})
```

A list of the genes encoded in the model can be accessed via `model.genes`.

```
In [14]: model.genes[0:10]
```

```
Out[14]: [<Gene b2215 at 0x10c6f3780>,
          <Gene b1377 at 0x10950b4e0>,
          <Gene b0241 at 0x109351be0>,
          <Gene b0929 at 0x109351048>,
          <Gene b4035 at 0x109351d68>,
          <Gene b4033 at 0x109344b38>,
          <Gene b4034 at 0x115e60518>,
          <Gene b4032 at 0x115e60550>,
          <Gene b4036 at 0x115e60588>,
          <Gene b4213 at 0x115e605c0>]
```

A few additional attributes have been added that are not available in a cobrapy model. For example, exchange reactions that allow certain metabolites to enter or leave the model can be accessed through `model.exchanges`.

```
In [15]: model.exchanges[0:10]
```

```
Out[15]: [<Reaction DM_4crsol_c at 0x115f44390>,
          <Reaction DM_5drib_c at 0x115f443c8>,
          <Reaction DM_aacald_c at 0x115f44400>,
          <Reaction DM_amob_c at 0x115f44438>,
          <Reaction DM_mththf_c at 0x115f44470>,
          <Reaction DM_oxam_c at 0x115f444a8>,
          <Reaction EX_12ppd__R_e at 0x115f44550>,
          <Reaction EX_12ppd__S_e at 0x115f44588>,
          <Reaction EX_14glucan_e at 0x115f445c0>,
          <Reaction EX_15dap_e at 0x115f445f8>]
```

Or, the current medium can be accessed through `model.medium`.

```
In [16]: model.medium.T
```

```
Out[16]:                 bound
         EX_ca2_e      1000.00
         EX_cbl1_e        0.01
         EX_cl_e       1000.00
         EX_co2_e      1000.00
         EX_cobalt2_e  1000.00
         ...                ...
         EX_sel_e      1000.00
         EX_slnt_e     1000.00
         EX_so4_e      1000.00
         EX_tungs_e    1000.00
         EX_zn2_e      1000.00

         [25 rows x 1 columns]
```

It is also possible to get a list of essential reactions ...

```
In [17]: from cameo.flux_analysis.analysis import find_essential_reactions
         find_essential_reactions(model)[0:10]

Out[17]: [<Reaction DM_4crsol_c at 0x115f44390>,
          <Reaction DM_5drib_c at 0x115f443c8>,
          <Reaction DM_amob_c at 0x115f44438>,
          <Reaction DM_mththf_c at 0x115f44470>,
          <Reaction BIOMASS_Ec_iJO1366_core_53p95M at 0x115f44518>,
          <Reaction EX_ca2_e at 0x115f5c3c8>,
          <Reaction EX_cl_e at 0x115f5c588>,
          <Reaction EX_cobalt2_e at 0x115f5c668>,
          <Reaction EX_cu2_e at 0x115f5c860>,
          <Reaction EX_glc__D_e at 0x115f697b8>]
```

... and essential genes.

```
In [18]: from cameo.flux_analysis.analysis import find_essential_genes
         find_essential_genes(model)[0:10]

Out[18]: [<Gene b4245 at 0x115e90048>,
          <Gene b0109 at 0x115f08080>,
          <Gene b2838 at 0x115ea80f0>,
          <Gene b0423 at 0x115f380f0>,
          <Gene b2574 at 0x115e90128>,
          <Gene b3809 at 0x115ea8128>,
          <Gene b4407 at 0x115f38128>,
          <Gene b0175 at 0x115ea8160>,
          <Gene b3992 at 0x115f38160>,
          <Gene b0928 at 0x115e90198>]
```

02-import-models

August 13, 2017

1 Import models

1.1 Import models from files

The function `~cameo.io.load_model` accepts a number of different file formats like SBML (Systems Biology Markup Language) for example.

```
In [1]: less data/e_coli_core.xml
```

```
In [2]: from cameo import load_model
        model = load_model('data/e_coli_core.xml')
```

```
In [3]: model
```

```
Out[3]: <SolverBasedModel e_coli_core at 0x10704c240>
```

Other accepted formats include pickle (simply pickled models) and json.

1.2 Import models from the internet

In the previous quick start chapter we demonstrated how to use `~cameo.io.load_model` to import a model by ID. But where did the model come from? Cameo has currently access to two model repositories on the internet, http://bigg.ucsd.edu and http://darwin.di.uminho.pt/models.

```
In [4]: from cameo import models
```

```
In [5]: models.index_models_bigg()
```

```
Out[5]:         bigg_id  gene_count  metabolite_count  \
        0   e_coli_core         137                72
        1   iAB_RBC_283         346               342
        2       iAF1260        1261              1668
        3      iAF1260b        1261              1668
        ..          ...         ...               ...
        76       iYO844         844               990
        77      iZ_1308        1308              1923
        78       RECON1        1905              2766
        79     STM_v1_0        1271              1800
```

```
                                          organism  reaction_count
0          Escherichia coli str. K-12 substr. MG1655              95
1                                       Homo sapiens             469
2          Escherichia coli str. K-12 substr. MG1655            2382
3          Escherichia coli str. K-12 substr. MG1655            2388
..                                               ...             ...
76       Bacillus subtilis subsp. subtilis str. 168            1250
77             Escherichia coli O157:H7 str. EDL933            2722
78                                      Homo sapiens            3742
79  Salmonella enterica subsp. enterica serovar Ty...            2545

[80 rows x 5 columns]
```

In [6]:
```
models.index_models_minho()
```

Out[6]:
```
      id           name                            doi              author  \
0      1         iJR904       10.1186/gb-2003-4-9-r54               Reed
1      2        iAF1260            10.1038/msb4100155              Feist
2      3         iMM904        10.1186/1752-0509-3-37                 Mo
3      4         iJP815  10.1371/journal.pcbi.1000210           Puchalka
..   ...            ...                           ...                ...
144  149         iYali4         10.1038/npjsba.2016.5           Kerkhoven
145  150  iLB1027_lipid  10.1371/journal.pone.0155038   Jennifer Levering
146  151        iLB1027  10.1371/journal.pone.0155038   Jennifer Levering
147  152        iMT1174     10.1186/s12918-015-0190-y  Mohammad Tajparast

     year  formats                                organism  \
0    2003   [sbml]  Escherichia coli str. K12 substr. MG1655
1    2007   [sbml]  Escherichia coli str. K12 substr. MG1655
2    2007   [sbml]                  Saccharomyces cerevisiae
3    2008   [sbml]            Pseudomonas putida str. KT2440
..    ...      ...                                       ...
144  2016   [sbml]                     Yarrowia lipolytica
145  2016   [sbml]                 Phaeodactylum tricornutum
146  2016   [sbml]                 Phaeodactylum tricornutum
147  2015  [excel]                   Rhodococcus jostii RHA1

                                          taxonomy validated
0    Bacteria; Proteobacteria; Gammaproteobacteria;...      True
1    Bacteria; Proteobacteria; Gammaproteobacteria;...      True
2    Eukaryota; Opisthokonta; Fungi; Dikarya; Ascom...      True
3    Bacteria; Proteobacteria; Gammaproteobacteria;...      True
..                                                 ...       ...
144  Eukaryota; Fungi; Dikarya; Ascomycota; Sacchar...     False
145  Eukaryota; Stramenopiles; Bacillariophyta; Bac...      True
146  Eukaryota; Stramenopiles; Bacillariophyta; Bac...      True
147  Bacteria; Terrabacteria group; Actinobacteria;...     False

[148 rows x 9 columns]
```

Models from BiGG and the University of Minho can conveniently be accessd via `~cameo.models.bigg` and `~cameo.models.minho` respectively.

```
In [7]: models.bigg.iJN746
```

```
Out[7]: <SolverBasedModel iJN746 at 0x1142a48d0>
```

```
In [8]: models.minho.iMM904
```

```
Out[8]: <SolverBasedModel iMM904 at 0x117f640b8>
```

Models in Minho database have been manually verified if they can be used to run simulations as described in the publications.

```
In [9]: models.minho.validated.VvuMBEL943 # use TAB completion to see the other models
```

```
Out[9]: <SolverBasedModel HyunUkKim2010_VvuMBEL943_MetabolicModeling at 0x115769cc0>
```

03-simulate-models

August 13, 2017

1 Simulate models

cameo uses and extends the model data structures defined by cobrapy, our favorite COnstraints-Based Reconstruction and Analysis tool for Python. cameo is thus 100% compatible with cobrapy. For efficiency reasons, however, cameo implements its own simulation methods that take advantage of a more advanced solver interface.

1.1 Primer: Constraint-Based Modeling

Constraint-based modeling is a powerful modeling framework for analyzing metabolism on the genome scale (McCloskey et al., 2013). For a model that encompasses n reactions that involve m metabolites, $\mathbf{S}$ is a matrix of dimension $m \times n$ that encodes the stoichiometry of the metabolic reaction system; it is usually referred to as stoichiometric matrix. Assuming that the system is in a steady state—the concentration of metabolites are constant—the system of flux-balances can be formulated as

$$\mathbf{S}\mathbf{v} = 0\,,$$

where $\mathbf{v}$ is the vector of flux rates. With the addition of a biologically meaningful objective, flux capacity constraints, information about the reversibility of reactions under physiological conditions, an optimization problem can be formulated that can easily be solved using linear programming.

, e.g., maximimization of biomass production,Given the maximization of growth rate as one potential biological objective $v_{biomass}$, i.e., the flux of an artificial reaction that consumes biomass components in empirically determined proportions, and assuming that the cell is evolutionary optimized to achieve that objective, and incorporating knowledge about reaction reversibility, uptake and secretion rates, and maximum flux capacities in the form of lower and uppers bounds ($\mathbf{v}_{lb}$ and $\mathbf{v}_{ub}$) on the flux variables $\mathbf{v}$, one can formulate and solve an optimization problem to identify an optimal set of flux rates using flux balance analysis (FBA):

$$\begin{aligned} Max \;\; & Z_{obj} = \mathbf{c}^T\mathbf{v} \\ \text{s.t.} \;\; & \mathbf{S}\mathbf{v} = 0 \\ & \mathbf{v}_{lb} \leq \mathbf{v} \leq \mathbf{v}_{ub}\,. \end{aligned}$$

1.2 Flux Balance Analysis

Load a model.

```
In [1]: from cameo import load_model
        model = load_model('iJO1366')
```

In cameo, flux balance analysis can be performed with the function `fba`.

```
In [2]: from cameo import fba
        %time fba_result = fba(model)

CPU times: user 141 ms, sys: 4.53 ms, total: 146 ms
Wall time: 145 ms
```

Basically, `fba` calls `model.solve()` and wraps the optimization solution in a `FluxDistributionResult` object. The maximum objective values (corresponding to a maximum growth rate) can obtained throug `result.objective_value`.

```
In [3]: fba_result.data_frame

Out[3]:                    flux
        DM_4crsol_c  2.1907e-04
        DM_5drib_c   2.2103e-04
        DM_aacald_c -0.0000e+00
        DM_amob_c    1.9647e-06
        DM_mththf_c  4.4010e-04
        ...                 ...
        ZN2abcpp     0.0000e+00
        ZN2t3pp      0.0000e+00
        ZN2tpp       3.3499e-04
        ZNabcpp      0.0000e+00
        Zn2tex       3.3499e-04

        [2583 rows x 1 columns]
```

Flux distributions can be visualized using *escher* :

```
In [4]: fba_result.display_on_map("iJO1366.Central metabolism")

<IPython.core.display.HTML object>
```

1.3 Parsimonious Flux Balance Analysis

Parsimonious flux balance analysis (Lewis et al., 2010), a variant of FBA, performs FBA in in a first step to determine the maximum objective value Z_{obj}, fixes it in form of an additional model constraint ($\mathbf{c}^T\mathbf{v} \geq Z_{obj}$), and then minimizes in a second optimization the L_1 norm of $\mathbf{v}$. The assumption behind pFBA is that cells try to minimize flux magnitude as well in order to keep protein costs low.

$$
\begin{aligned}
Max \quad & |\mathbf{v}| \\
\text{s.t.} \quad & \mathbf{S}\mathbf{v} = 0 \\
& \mathbf{c}^T\mathbf{v} \geq Z_{obj} \\
& \mathbf{v}_{lb} \leq \mathbf{v} \leq \mathbf{v}_{ub} \,.
\end{aligned}
$$

In cameo, pFBA can be performed with the function `pfba`.

```
In [5]: from cameo import pfba
        %time pfba_result = pfba(model)
```

```
CPU times: user 551 ms, sys: 9.12 ms, total: 560 ms
Wall time: 604 ms
```

The `objective_function` value is $|\mathbf{v}|$ ...

```
In [6]: pfba_result.objective_value
```

```
Out[6]: 699.0222751839508
```

... which is smaller than flux vector of the original FBA solution.

```
In [7]: abs(fba_result.data_frame.flux).sum()
```

```
Out[7]: 763.15667411330537
```

1.4 Setp 2: Simulate knockouts phenotypes

Although PFBA and FBA can be used to simulate the effect of knockouts, other methods have been proven more valuable for that task: MOMA and ROOM. In *cameo* we implement a linear version of MOMA.

Simulating knockouts:

- Manipulate the bounds of the reaction (or use the shorthand method knock_out)

```
In [8]: model.reactions.PGI
```

```
Out[8]: <Reaction PGI at 0x11324ff28>
```

```
In [9]: model.reactions.PGI.knock_out()
        model.reactions.PGI
```

```
Out[9]: <Reaction PGI at 0x11324ff28>
```

- Simulate using different methods:

```
In [10]: %time fba_knockout_result = fba(model)
         fba_knockout_result[model.reactions.BIOMASS_Ec_iJO1366_core_53p95M]

CPU times: user 47.7 ms, sys: 2.61 ms, total: 50.3 ms
Wall time: 51.2 ms


Out[10]: 0.9761293262947276

In [11]: %time pfba_knockout_result = pfba(model)
         pfba_knockout_result[model.reactions.BIOMASS_Ec_iJO1366_core_53p95M]

CPU times: user 535 ms, sys: 4.16 ms, total: 539 ms
Wall time: 559 ms


Out[11]: 0.9761293262947276
```

MOMA and ROOM relly on a reference (wild-type) flux distribution and we can use the one previously computed.

Parsimonious FBA references seem to produce better results using this methods

```
In [12]: from cameo.flux_analysis.simulation import room, lmoma

In [13]: %time lmoma_result = lmoma(model, reference=pfba_result.fluxes)
         lmoma_result[model.reactions.BIOMASS_Ec_iJO1366_core_53p95M]

CPU times: user 25.2 s, sys: 150 ms, total: 25.4 s
Wall time: 25.6 s


Out[13]: 0.87240935362436134
```

ROOM is a dificult computational problem. If the bounds of the system are not large enought, it can take many hours to simulate. To improve the speed of the simulation and the chances of finding a solution, we increase the bounds.

```
In [14]: for reaction in model.reactions:
             if reaction.upper_bound == 1000:
                 reaction.upper_bound = 99999999
             if reaction.lower_bound == -1000:
                 reaction.lower_bound = -99999999

In [15]: %time room_result = room(model, reference=pfba_result.fluxes)
         room_result[model.reactions.BIOMASS_Ec_iJO1366_core_53p95M]

CPU times: user 17.5 s, sys: 105 ms, total: 17.6 s
Wall time: 17.3 s


Out[15]: 0.95190065834517257
```

04-analyze-models

August 13, 2017

1 Analyzing models

cameo uses and extends the model data structures defined by cobrapy, our favorite **CO**nstraints-**B**ased **R**econstruction and **A**nalysis tool for **Py**thon. **cameo** is thus 100% compatible with **cobrapy**. For efficiency reasons though **cameo** implements its own analysis methods that take advantage of a more advanced solver interface.

```
In [1]: from cameo import models
        model = models.bigg.e_coli_core
```

1.1 Flux Variability Analysis

Flux variability analysis (FVA) enables the computation of lower and upper bounds of reaction fluxes.

```
In [2]: from cameo import flux_variability_analysis
```

```
In [3]: fva_result = flux_variability_analysis(model)
        fva_result.data_frame
```

```
Out[3]:         lower_bound  upper_bound
        ACALD       -20.0000         0.00
        ACALDt      -20.0000         0.00
        ACKr        -20.0000         0.00
        ACONTa        0.0000        20.00
        ACONTb        0.0000        20.00
        ...              ...          ...
        TALA         -0.1545        20.00
        THD2          0.0000       333.22
        TKT1         -0.1545        20.00
        TKT2         -0.4664        20.00
        TPI         -10.0000        10.00

        [95 rows x 2 columns]
```

```
In [4]: fva_result.plot(index=fva_result.data_frame.index[:25])
```

One very useful application of FVA is determining if alternative optimal solution exist.

```
In [5]: fva_result2 = flux_variability_analysis(model,fraction_of_optimum=0.5)
        fva_result2.data_frame
```

```
Out[5]:          lower_bound  upper_bound
        ACALD       -12.6025       0.0000
        ACALDt      -12.6025       0.0000
        ACKr        -13.3589       0.0000
        ACONTa        0.4714      13.8303
        ACONTb        0.4714      13.8303
        ...              ...          ...
        TALA         -0.1545      13.2807
        THD2          0.0000     168.6767
        TKT1         -0.1545      13.2807
        TKT2         -0.4664      13.1229
        TPI          -3.7935       9.5654

        [95 rows x 2 columns]
```

```
In [6]: fva_result2.plot()
```

```
In [7]: from cameo.visualization import plotting
```

1.2 Phenotypic Phase Plane

The phenotypic phase plane is a modeling technique was developed to do a theoretical accessement of what cell can or cannot do in terms of the stoichiometric constraints [Edawards *et al.* 2001].

The phenotipic phase plane between growth and a product of interest yields the production envelope: a representation between the trade of between production of the desired product and growth.

```
In [8]: from cameo import phenotypic_phase_plane
```

```
In [9]: model.reactions.EX_o2_e.lower_bound = -10
        result = phenotypic_phase_plane(model,
                                        variables=[model.reactions.BIOMASS_Ecoli_core_w_GAM],
                                        objective=model.reactions.EX_succ_e,
                                        points=10)
```

```
In [10]: result.plot()
```

The production envelope allows is a quick way to inspect the limitations of the system to design and how the production relates for with growth. In the previous example, succinate prudction is completly decoupled from growth and by decreasing the growth rate it is theoretically possible to produce up to 15 times more succinate.

```
In [11]: result.plot(points=[(0.52, 0), (0.23, 12.2)], points_colors=["green", "red"])
```

The production envelope can show the coupling between growth and production. There is no stoichiometric couple between growth and production for succinate under aerobic conditions, but that is not the case for acetate under anaerobic conditions.

```
In [12]: result = phenotypic_phase_plane(model,
                                         variables=[model.reactions.BIOMASS_Ecoli_core_w_GAM],
                                         objective=model.reactions.EX_ac_e)
         result.plot()

In [13]: result.data_frame
```

Out[13]:

	BIOMASS_Ecoli_core_w_GAM	objective_lower_bound	objective_upper_bound
0	0.0000	0.0000	20.0000
1	0.0294	0.0000	19.5528
2	0.0588	0.0000	19.1056
3	0.0883	0.0000	18.6584
4	0.1177	0.0000	18.2112
..	...	...	...
15	0.4414	0.0000	13.2921
16	0.4708	0.0000	12.8449
17	0.5002	0.0000	12.3977
18	0.5296	2.6280	11.9505
19	0.5591	9.9057	11.5033

	c_yield_lower_bound	c_yield_upper_bound	mass_yield_lower_bound
0	0.0000	0.6667	0.0000
1	0.0000	0.6518	0.0000
2	0.0000	0.6369	0.0000
3	0.0000	0.6219	0.0000
4	0.0000	0.6070	0.0000
..	...	...	...
15	0.0000	0.4431	0.0000
16	0.0000	0.4282	0.0000
17	0.0000	0.4133	0.0000
18	0.0876	0.3983	0.0861
19	0.3302	0.3834	0.3246

	mass_yield_upper_bound
0	0.6555
1	0.6408
2	0.6262
3	0.6115
4	0.5969
..	...
15	0.4356
16	0.4210
17	0.4063
18	0.3917
19	0.3770

[20 rows x 7 columns]

One can imediatly see if a design is feasible within the new defined constraints.

```
In [14]: result.plot(points=[(0.2, 8), (0.2, 2)], points_colors=["green", "red"])
```

The computed data can be inspected in the format of a pandas data frame by calling `result.data_frame`

```
In [15]: result.data_frame
```

```
Out[15]:     BIOMASS_Ecoli_core_w_GAM  objective_lower_bound  objective_upper_bound  \
         0                    0.0000                 0.0000                20.0000
         1                    0.0294                 0.0000                19.5528
         2                    0.0588                 0.0000                19.1056
         3                    0.0883                 0.0000                18.6584
         4                    0.1177                 0.0000                18.2112
         ..                      ...                    ...                    ...
         15                   0.4414                 0.0000                13.2921
         16                   0.4708                 0.0000                12.8449
         17                   0.5002                 0.0000                12.3977
         18                   0.5296                 2.6280                11.9505
         19                   0.5591                 9.9057                11.5033

             c_yield_lower_bound  c_yield_upper_bound  mass_yield_lower_bound  \
         0                0.0000               0.6667                  0.0000
         1                0.0000               0.6518                  0.0000
         2                0.0000               0.6369                  0.0000
         3                0.0000               0.6219                  0.0000
         4                0.0000               0.6070                  0.0000
         ..                  ...                  ...                     ...
         15               0.0000               0.4431                  0.0000
         16               0.0000               0.4282                  0.0000
         17               0.0000               0.4133                  0.0000
         18               0.0876               0.3983                  0.0861
         19               0.3302               0.3834                  0.3246

             mass_yield_upper_bound
         0                   0.6555
         1                   0.6408
         2                   0.6262
         3                   0.6115
         4                   0.5969
         ..                     ...
         15                  0.4356
         16                  0.4210
         17                  0.4063
         18                  0.3917
         19                  0.3770

         [20 rows x 7 columns]
```

```
In [16]: model.reactions.EX_o2_e.lower_bound = 0
         result2 = phenotypic_phase_plane(model,
```

```
                                        variables=[model.reactions.BIOMASS_Ecoli_core_w_GAM]
                                        objective=model.reactions.EX_ac_e,
                                        points=10)
        result2.plot()
```

1.3 Flux Balance Impact Degree

```
In [17]: from cameo.flux_analysis.analysis import flux_balance_impact_degree

In [18]: model.reactions.EX_o2_e.lower_bound = -10

In [19]: %time fbid = flux_balance_impact_degree(model, ["EX_o2_e"])

CPU times: user 287 ms, sys: 4.74 ms, total: 292 ms
Wall time: 294 ms


In [20]: fbid

Out[20]: <cameo.flux_analysis.analysis.FluxBalanceImpactDegreeResult at 0x1056230b8>
```

05-predict-gene-knockout-strategies

September 4, 2017

1 Predict gene knockout strategies

In cameo we have two ways of predicting gene knockout targets: using evolutionary algorithms (OptGene) or linear programming (OptKnock)

If you're running this notebook on try.cameo.bio, things might run very slow due to our inability to provide access to the proprietary CPLEX solver on a public webserver. Furthermore, Jupyter kernels might crash and restart due to memory limitations on the server.

```
In [1]: from cameo import models
```

```
In [2]: model = models.bigg.iJO1366
```

```
In [3]: wt_solution = model.optimize()
        growth = wt_solution.fluxes["BIOMASS_Ec_iJO1366_core_53p95M"]
        acetate_production = wt_solution.fluxes["EX_ac_e"]
```

```
In [4]: from cameo import phenotypic_phase_plane
```

```
In [5]: p = phenotypic_phase_plane(model, variables=['BIOMASS_Ec_iJO1366_core_53p95M'], objectiv
        p.plot(points=[(growth, acetate_production)])
```

1.1 OptGene

OptGene is an approach to search for gene or reaction knockouts that relies on evolutionary algorithms[1]. The following image from authors summarizes the OptGene workflow.

Every iteration we keep the best 50 individuals so we can generate a library of targets.

```
In [6]: from cameo.strain_design.heuristic.evolutionary_based import OptGene
```

```
In [7]: optgene = OptGene(model)
```

```
In [8]: result = optgene.run(target=model.reactions.EX_ac_e,
                            biomass=model.reactions.BIOMASS_Ec_iJO1366_core_53p95M,
                            substrate=model.metabolites.glc__D_e,
                            max_evaluations=5000,
                            plot=False)
```

```
Starting optimization at Tue, 18 Jul 2017 17:55:41
```

```
Finished after 05:08:36
```

```
In [9]: result
```

```
Out[9]: <cameo.strain_design.heuristic.evolutionary_based.OptGeneResult at 0x116fc6198>
```

```
In [10]: result.plot(0)
```

```
In [11]: result.display_on_map(0, "iJO1366.Central metabolism")
```

```
<IPython.core.display.HTML object>
```

1.2 OptKnock

OptKnock uses a bi-level mixed integer linear programming approach to identify reaction knockouts[2]:

$$\begin{aligned} \max_{\mathbf{y}}: \quad & v_{chemical} && \textbf{(OptKnock)} \\ \text{s.t.} \quad & \left[\begin{array}{ll} \max: & \mathbf{c}^T \cdot \mathbf{v} \\ \text{s.t.} & \mathbf{S} \cdot \mathbf{v} = 0 \\ & v_{biomass} \geq v_{min\_biomass} \\ & \mathbf{v}_{lb} \cdot \mathbf{y} \leq \mathbf{v} \leq \mathbf{v}_{ub} \cdot \mathbf{y} \end{array}\right] && \textbf{(Primal)} \\ & \mathbf{y} = \{0,1\} \\ & \|\mathbf{y}\|_1 \leq K \end{aligned}$$

```
In [12]: from cameo.strain_design.deterministic.linear_programming import OptKnock
```

```
In [13]: optknock = OptKnock(model, fraction_of_optimum=0.1)
```

Running multiple knockouts with OptKnock can take a few hours or days...

```
In [14]: result = optknock.run(max_knockouts=1, target="EX_ac_e", biomass="BIOMASS_Ec_iJO1366_co
```

```
<IPython.core.display.HTML object>
```

```
<IPython.core.display.Javascript object>
```

```
In [15]: result
```

```
Out[15]: <cameo.strain_design.deterministic.linear_programming.OptKnockResult at 0x11ad3b828>
```

```
In [16]: result.plot(0)
```

```
In [17]: result.display_on_map(0, "iJO1366.Central metabolism")
```

```
<IPython.core.display.HTML object>
```

1.3 References

[1]Patil, K. R., Rocha, I., Förster, J., & Nielsen, J. (2005). Evolutionary programming as a platform for in silico metabolic engineering. BMC Bioinformatics, 6, 308. doi:10.1186/1471-2105-6-308

[2]Burgard, A.P., Pharkya, P., Maranas, C.D. (2003), "OptKnock: A Bilevel Programming Framework for Identifying Gene Knockout Strategies for Microbial Strain Optimization," Biotechnology and Bioengineering, 84(6), 647-657.

06-predict-gene-modulation-targets

September 4, 2017

1 Predict expression modulation targets

Cameo provides algorithms to search for genes or reactions that can be over or down regulated in order to achieve a given biological objective.

```
In [1]: from cameo import models
```

Load the E. coli core model.

```
In [2]: model = models.bigg.e_coli_core
```

1.1 Flux Scanning based on Enforced Objective Flux

```
In [3]: from cameo.strain_design.deterministic.flux_variability_based import FSEOF

In [4]: fseof = FSEOF(model)

In [5]: fseof.run(target=model.reactions.EX_succ_e)

Out[5]: <cameo.strain_design.deterministic.flux_variability_based.FSEOFResult at 0x1151bc978>
```

1.2 Differential flux variability analysis

Compares flux ranges of a reference model to a set of models that have been parameterized to lie on a grid of evenly spaced points in the n-dimensional production envelope (n being the number of reaction bounds to be varied).

```
In [6]: from cameo.flux_analysis.analysis import phenotypic_phase_plane
        from cameo.strain_design.deterministic import DifferentialFVA
```

1.2.1 Succinate production

The production envelope looks like this.

```
In [7]: production_envelope = phenotypic_phase_plane(model,
                                                     variables=[model.reactions.BIOMASS_Ecoli_co
                                                     objective=model.metabolites.succ_e)
        production_envelope.plot(height=400)
```

Set up a model that represents a reference state (in this case a model with a constrained growth rate).

```
In [8]: model.reactions.EX_o2_e.lower_bound = 0
        reference_model = model.copy()
        biomass_rxn = reference_model.reactions.BIOMASS_Ecoli_core_w_GAM
        biomass_rxn.lower_bound = 0.
        target = reference_model.metabolites.succ_e
```

Set up the differential flux variability analysis strain design method.

```
In [9]: diffFVA = DifferentialFVA(design_space_model=model,
                                  reference_model=reference_model,
                                  objective=target,
                                  variables=[biomass_rxn],
                                  normalize_ranges_by=biomass_rxn,
                                  points=10)
```

Run differential flux variability analysis (only on the surface of the production envelope)

```
In [10]: result = diffFVA.run(surface_only=True)

In [11]: result.solutions
```

Out[11]:

reaction	lower_bound	upper_bound	gaps	normalized_gaps	biomass	production	KO
ACALD	0.000000	0.000000	-2.339592e+00	NaN	0.000000	13.905778	True
ACALDt	0.000000	0.000000	0.000000e+00	NaN	0.000000	13.905778	False
ACKr	-5.664889	-5.664889	0.000000e+00	inf	0.000000	13.905778	False
ACONTa	0.429333	0.429333	0.000000e+00	inf	0.000000	13.905778	False
ACONTb	0.429333	0.429333	0.000000e+00	inf	0.000000	13.905778	False
...	...	...	...	...	...	...	...
TALA	-0.033659	-0.033659	0.000000e+00	-0.141033	0.188145	2.123333	False
THD2	3.225950	3.225950	0.000000e+00	4.869204	0.188145	2.123333	False
TKT1	-0.033659	-0.033659	0.000000e+00	-0.141033	0.188145	2.123333	False
TKT2	-0.101579	-0.101579	0.000000e+00				
TPI	9.812852	9.812852	0.000000e+00				

```
TKT2                          -0.425623  0.188145    2.123333  False
TPI                           42.313745  0.188145    2.123333  False

                            flux_reversal  suddenly_essential  free_flux  \
reaction
ACALD                               False               False      False
ACALDt                              False               False      False
ACKr                                False               False      False
ACONTa                              False               False      False
ACONTb                              False               False      False
...                                   ...                 ...        ...
TALA                                False               False      False
THD2                                False               False      False
TKT1                                False               False      False
TKT2                                False               False      False
TPI                                 False               False      False

                                        reaction  excluded
reaction
ACALD                                     ACALD     False
ACALDt                                   ACALDt     False
ACKr                                       ACKr     False
ACONTa                                   ACONTa     False
ACONTb                                   ACONTb     False
...                                         ...       ...
TALA                                       TALA     False
THD2                                       THD2     False
TKT1                                       TKT1     False
TKT2                                       TKT2     False
TPI                                         TPI     False

[684 rows x 12 columns]
```

In [12]:
```
result.plot(5, variables=['FBP', 'G6PDH2r', 'PGL', 'PGK'])
```

In [13]:
```
result.display_on_map(2, map_name="iJO1366.Central metabolism")
```

```
<IPython.core.display.HTML object>
```

In [ ]:

07-predict-heterologous-pathways

August 13, 2017

```
In [1]: from IPython.display import display
        import re
```

1 Predict heterologous pathways

Predicting heterologous pathways is an important strategy to generate new viable strains. Because portfolio of available reactions is very large, computer assisted pathway design becomes essential. **Cameo** implements a pathway search algorithm using an universal biochemical reaction database that enumerates the shortest pathways.

If you're running this notebook on try.cameo.bio, things might run very slow due to our inability to provide access to the proprietary CPLEX solver on a public webserver. Furthermore, Jupyter kernels might crash and restart due to memory limitations on the server.

```
In [2]: from cameo import models
        from cameo.strain_design import pathway_prediction

In [3]: model = models.bigg.iMM904

In [4]: predictor = pathway_prediction.PathwayPredictor(model)

In [5]: pathways = predictor.run(product="vanillin", max_predictions=4)
```

```
<IPython.core.display.HTML object>
```

```
                                                  equation  lower_bound  \
MNXR5340  H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336  2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR230   H(+) + 4-hydroxybenzoate + O2 + NADPH <=> H2O ...        -1000

          upper_bound
MNXR5340         1000
MNXR5336         1000
MNXR230          1000


Max flux: 1.90533
```

<IPython.core.display.HTML object>

	equation	lower_bound	upper_bound
MNXR5340	H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...	-1000	1000
MNXR5336	2.0 H(+) + NADH + vanillate <=> H2O + vanillin...	-1000	1000
MNXR68718	H2O + 3,4-dihydroxybenzoate <=> 3-dehydroshiki...	-1000	1000

Max flux: 3.36842

<IPython.core.display.HTML object>

	equation	lower_bound	upper_bound
MNXR4008	H(+) + 3-oxoadipate <=> H2O + 5-oxo-4,5-dihydr...	-1000	1000
MNXR184	3-oxoadipyl-CoA + succinate <=> 3-oxoadipate +...	-1000	1000
MNXR5340	H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...	-1000	1000
MNXR5336	2.0 H(+) + NADH + vanillate <=> H2O + vanillin...	-1000	1000
MNXR228	CO2 + 5-oxo-4,5-dihydro-2-furylacetate <=> H(+...	-1000	1000
MNXR4119	2.0 H(+) + 3-carboxy-cis,cis-muconate <=> 3,4-...	-1000	1000
MNXR209	CoA + 3-oxoadipyl-CoA <=> acetyl-CoA + succiny...	-1000	1000
MNXR3655	2-(carboxymethyl)-5-oxo-2,5-dihydro-2-furoate ...	-1000	1000

Max flux: 5.59223

<IPython.core.display.HTML object>

	equation	lower_bound	upper_bound
MNXR5338	2.0 H(+) + NADH + 3,4-dihydroxybenzoate <=> H2...	-1000	1000

```
MNXR1041   diphosphate + AMP + caffeoyl-CoA <=> CoA + ATP...        -1000
MNXR4974   O2 + 2.0 trans-4-coumarate <=> 2.0 trans-caffeate       -1000
MNXR227    diphosphate + AMP + 4-coumaroyl-CoA <=> CoA + ...       -1000
MNXR5340   H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...       -1000
MNXR5336   2.0 H(+) + NADH + vanillate <=> H2O + vanillin...       -1000
MNXR18369  CoA + H2O + 4-coumaroyl-CoA + NAD(+) <=> H(+) ...       -1000
MNXR232    H(+) + CoA + 4-hydroxybenzoate <=> H2O + 4-hyd...       -1000
MNXR1039   acetyl-CoA + 3,4-dihydroxybenzaldehyde <=> H2O...       -1000

           upper_bound
MNXR5338          1000
MNXR1041          1000
MNXR4974          1000
MNXR227           1000
MNXR5340          1000
MNXR5336          1000
MNXR18369         1000
MNXR232           1000
MNXR1039          1000


Max flux: 2.24390


In [6]: pathways.pathways[0].reactions[0]

Out[6]: <Reaction MNXR5340 at 0x1227016d8>

In [7]: pathways.plot_production_envelopes(model, objective=model.reactions.BIOMASS_SC5_notrac

This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]
[ (2,1) x3,y3 ]  [ (2,2) x4,y4 ]
```

08-high-level-API

August 13, 2017

1 Easy strain design using a high-level interface

WARNING: if you're running this notebook on try.cameo.bio, things might run very slow due to our inability to provide access to the CPLEX solver on a public webserver. Furthermore, Jupyter kernels might crash and restart due to memory limitations on the server.

Users primarily interested in using cameo as a tool for enumerating metabolic engineering strategies have access to cameo's advanced programming interface via `cameo.api` that provides access to potential products (`cameo.api.products`), host organisms (`cameo.api.hosts`) and a configurable design function (`cameo.api.design`). Running `cameo.api.design` requires only minimal input and will run the following workflow.

Import the advanced interface.

```
In [1]: from cameo import api
```

1.1 Searching for products

Search by trivial name.

```
In [2]: api.products.search('caffeine')
```

```
Out[2]:                                                    InChI  \
        MNXM680  InChI=1S/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)...

                                      SMILES charge    formula      mass      name  \
        MNXM680  CN1C=NC2=C1C(=O)N(C)C(=O)N2C      0  C8H10N4O2  194.1906  caffeine

                     source  search_rank
        MNXM680  chebi:27732            0
```

Search by ChEBI ID.

```
In [3]: api.products.search('chebi:27732')
```

```
Out[3]:                                                    InChI  \
        MNXM680  InChI=1S/C8H10N4O2/c1-10-4-9-6-5(10)7(13)12(3)...

                                      SMILES charge    formula      mass      name  \
        MNXM680  CN1C=NC2=C1C(=O)N(C)C(=O)N2C      0  C8H10N4O2  194.1906  caffeine
```

```
                source  search_rank
MNXM680  chebi:27732            0
```

1.2 Host organisms

Currently the following host organisms and respective models are available in cameo. More hosts and models will be added in the future (please get in touch with us if you'd like to get a particular host organism included).

```
In [4]: for host in api.hosts:
            for model in host.models:
                print(host.name, model.id)
```

```
Escherichia coli iJO1366
Saccharomyces cerevisiae iMM904
```

1.3 Computing strain engineering strategies

For demonstration purposes, we'll set a few options to limit the computational time. Also we'll create a multiprocessing view to take advantage of multicore CPUs (strain design algorithms will be run in parallel for individually predicted heterologous pathways).

```
In [5]: from cameo.parallel import MultiprocessingView
        mp_view = MultiprocessingView()
```

Limit the number of predicted heterlogous pathways to 4.

```
In [6]: api.design.options.max_pathway_predictions = 4
```

Set a time limit of 30 minutes on individual heuristic optimizations.

```
In [7]: api.design.options.heuristic_optimization_timeout = 30
```

```
In [ ]: report = api.design(product='vanillin', view=mp_view)
```

```
<IPython.core.display.HTML object>


<IPython.core.display.HTML object>


<IPython.core.display.HTML object>


<IPython.core.display.HTML object>


<IPython.core.display.HTML object>
```

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

<IPython.core.display.HTML object>

```
                                                    equation  lower_bound  \
MNXR5340   H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336   2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR68718  H2O + 3,4-dihydroxybenzoate <=> 3-dehydroshiki...        -1000
MNXR651    2.0 H(+) + NADH + formate <=> H2O + formaldehy...        -1000

           upper_bound
MNXR5340          1000
MNXR5336          1000
MNXR68718         1000
MNXR651           1000
```

Max flux: 7.58479

<IPython.core.display.HTML object>

```
                                                    equation  lower_bound  \
MNXR5340   H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336   2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR2795   S-adenosyl-L-methionine + glycine <=> H(+) + S...        -1000
MNXR68718  H2O + 3,4-dihydroxybenzoate <=> 3-dehydroshiki...        -1000

           upper_bound
MNXR5340          1000
MNXR5336          1000
MNXR2795          1000
MNXR68718         1000
```

Max flux: 4.29196

<IPython.core.display.HTML object>

```
                                                  equation  lower_bound  \
MNXR5340  H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
```

```
MNXR5336  2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR230   H(+) + 4-hydroxybenzoate + O2 + NADPH <=> H2O ...        -1000
MNXR640    methanol + NAD(+) <=> H(+) + NADH + formaldehyde        -1000

          upper_bound
MNXR5340         1000
MNXR5336         1000
MNXR230          1000
MNXR640          1000


<IPython.core.display.HTML object>


                                                    equation  lower_bound  \
MNXR5340   H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336   2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR68718  H2O + 3,4-dihydroxybenzoate <=> 3-dehydroshiki...        -1000
MNXR4427          methanol + H2O2 <=> 2.0 H2O + formaldehyde        -1000

           upper_bound
MNXR5340          1000
MNXR5336          1000
MNXR68718         1000
MNXR4427          1000


<IPython.core.display.Javascript object>


This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]



<IPython.core.display.HTML object>


<IPython.core.display.HTML object>


<IPython.core.display.HTML object>


                                                  equation  lower_bound  \
MNXR5340  H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336  2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR230   H(+) + 4-hydroxybenzoate + O2 + NADPH <=> H2O ...        -1000
```

```
          upper_bound
MNXR5340         1000
MNXR5336         1000
MNXR230          1000


Max flux: 1.90533


<IPython.core.display.HTML object>


                                                  equation  lower_bound  \
MNXR5340   H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336   2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR68718  H2O + 3,4-dihydroxybenzoate <=> 3-dehydroshiki...        -1000

           upper_bound
MNXR5340          1000
MNXR5336          1000
MNXR68718         1000


Max flux: 3.36842


<IPython.core.display.HTML object>


                                                 equation  lower_bound  \
MNXR4008  H(+) + 3-oxoadipate <=> H2O + 5-oxo-4,5-dihydr...        -1000
MNXR184   3-oxoadipyl-CoA + succinate <=> 3-oxoadipate +...        -1000
MNXR5340  H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336  2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR228   CO2 + 5-oxo-4,5-dihydro-2-furylacetate <=> H(+...        -1000
MNXR4119  2.0 H(+) + 3-carboxy-cis,cis-muconate <=> 3,4-...        -1000
MNXR209   CoA + 3-oxoadipyl-CoA <=> acetyl-CoA + succiny...        -1000
MNXR3655  2-(carboxymethyl)-5-oxo-2,5-dihydro-2-furoate ...        -1000

          upper_bound
MNXR4008         1000
MNXR184          1000
MNXR5340         1000
MNXR5336         1000
MNXR228          1000
MNXR4119         1000
MNXR209          1000
MNXR3655         1000
```

```
Max flux: 5.59223


<IPython.core.display.HTML object>


                                                    equation  lower_bound  \
MNXR5338   2.0 H(+) + NADH + 3,4-dihydroxybenzoate <=> H2...        -1000
MNXR1041   diphosphate + AMP + caffeoyl-CoA <=> CoA + ATP...        -1000
MNXR4974   O2 + 2.0 trans-4-coumarate <=> 2.0 trans-caffeate        -1000
MNXR227    diphosphate + AMP + 4-coumaroyl-CoA <=> CoA + ...        -1000
MNXR5340   H(+) + NADH + O2 + vanillate <=> H2O + 3,4-dih...        -1000
MNXR5336   2.0 H(+) + NADH + vanillate <=> H2O + vanillin...        -1000
MNXR18369  CoA + H2O + 4-coumaroyl-CoA + NAD(+) <=> H(+) ...        -1000
MNXR232    H(+) + CoA + 4-hydroxybenzoate <=> H2O + 4-hyd...        -1000
MNXR1039   acetyl-CoA + 3,4-dihydroxybenzaldehyde <=> H2O...        -1000

           upper_bound
MNXR5338          1000
MNXR1041          1000
MNXR4974          1000
MNXR227           1000
MNXR5340          1000
MNXR5336          1000
MNXR18369         1000
MNXR232           1000
MNXR1039          1000


Max flux: 2.24390


<IPython.core.display.Javascript object>


This is the format of your plot grid:
[ (1,1) x1,y1 ]  [ (1,2) x2,y2 ]
[ (2,1) x3,y3 ]  [ (2,2) x4,y4 ]



Optimizing 6 pathways


In [ ]: report
```

1.3.1 IPython notebook

Click here to download this page as an IPython notebook.

09-vanillin-production

August 13, 2017

0.1 Vanillin production

In 2010, Brochado *et al* used heuristic optimization together with flux simulations to design a vanillin producing yeast strain.

Brochado, A. R., Andrejev, S., Maranas, C. D., & Patil, K. R. (2012). Impact of stoichiometry representation on simulation of genotype-phenotype relationships in metabolic networks. PLoS Computational Biology, 8(11), e1002758. doi:10.1371/journal.pcbi.1002758

0.2 Genome-scale metabolic model

In their work, the authors used *iFF708* model, but recent insights in Yeast yielded newer and more complete versions. Becuase this algorithms should be agnostic to the model, we implement the same strategy with a newer model.

```
In [1]: from cameo import models
```

```
In [2]: model = models.bigg.iMM904.copy()
```

Constraints can be set in the model according to data found in the literature. The defined conditions allow the simulation of phenotypes very close to the experimental results.

Model validation by comparing in silico prediction of the specific growth rate with experimental data. Growth phenotypes were collected from literature and compared to simulated values for chemostat cultivations at four different conditions, nitrogen limited aerobic (green) and anaerobic (red), carbon limited aerobic (blue) and anaerobic (white).

Österlund, T., Nookaew, I., Bordel, S., & Nielsen, J. (2013). Mapping condition-dependent regulation of metabolism in yeast through genome-scale modeling. BMC Systems Biology, 7, 36. doi:10.1186/1752-0509-7-36

```
In [3]: model.reactions.EX_glc__D_e.lower_bound = -13 #glucose exchange
        model.reactions.EX_o2_e.lower_bound = -3  #oxygen exchange
```

```
In [4]: model.medium
```

```
Out[4]:        EX_fe2_e  EX_glc__D_e  EX_h2o_e    EX_h_e    EX_k_e  EX_na1_e  \
        bound  999999.0           13  999999.0  999999.0  999999.0  999999.0

               EX_nh4_e  EX_o2_e   EX_pi_e  EX_so4_e
        bound  999999.0        3  999999.0  999999.0
```

```
In [5]: model.objective = model.reactions.BIOMASS_SC5_notrace #growth
        model.optimize().objective_value

Out[5]: 0.3902223535079852
```

0.3 Heterologous pathway

Vanillin is not produced by *S. cervisiae*. In their work an heterolgous pathway is inserted to allow generate a vanillin production strain. The pathway is described as:

Schematic representation of the de novo VG biosynthetic pathway in S. Cerevisisae (as designed by Hansen et al [5]). Metabolites are shown in black, enzymes are shown in black and in italic, cofactors and additional precursors are shown in red. Reactions catalyzed by heterologously introduced enzymes are shown in red. Reactions converting glucose to aromatic amino acids are represented by dashed black arrows. Metabolite secretion is represented by solid black arrows where relative thickness corresponds to relative extracellular accumulation. 3-DSH stands for 3-dedhydroshikimate, PAC stands for protocathechuic acid, PAL stands for protocatechuic aldehyde, SAM stands for S-adenosylmethionine. 3DSD stands for 3-dedhydroshikimate dehydratase, ACAR stands for aryl carboxylic acid reductase, PPTase stands for phosphopantetheine transferase, hsOMT stands for O-methyltransferase, and UGT stands for UDP-glycosyltransferase. Adapted from Hansen et al. [5]. Brochado et al. Microbial Cell Factories 2010 9:84 doi:10.1186/1475-2859-9-84

Using **cameo**, is very easy to generate a pathway and add it to a model.

```
In [6]: from cameo.core.pathway import Pathway

In [7]: vanillin_pathway = Pathway.from_file("data/vanillin_pathway.tsv")
        vanillin_pathway.data_frame

Out[7]:                                                    equation  lower_bound  \
        3DSD         3-dehydroshikimate --> H2O + protocathechuic acid          0.0
        ACAR_PPTase  ATP + NADPH + protocathechuic acid --> ADP + N...          0.0
        hsOMT        S-adenosyl-L-methionine + protocatechuic aldeh...          0.0
        UGT            UDP-glucose + Vanillin --> vanillin-B-glucoside          0.0

                     upper_bound
        3DSD              1000.0
        ACAR_PPTase       1000.0
        hsOMT             1000.0
        UGT               1000.0
```

And now we can plug the pathway to the model.

```
In [8]: vanillin_pathway.plug_model(model)

In [9]: from cameo import phenotypic_phase_plane
```

The Phenotypic phase plane can be used to analyse the theoretical yields at different growth rates.

```
In [10]: production_envelope = phenotypic_phase_plane(model, variables=[model.reactions.BIOMASS
                                                        objective=model.reactions.EX_vnl_b_glu_c
         production_envelope.plot()
```

To find gene knockout targets, we use `cameo.strain_design.heuristic` package which implements the OptGene strategy.

The authors used the biomass-product coupled yield (bpcy) for optimization which is the equivalent of running OptGene in non-robust mode. All simulations were computed using MOMA we use it's equivalent linear version (minimizing the absolute distance instead of the quadratic distance). The linear MOMA version is faster than the original MOMA formulation.

```
In [11]: from cameo.strain_design.heuristic.evolutionary_based import OptGene
         from cameo.flux_analysis.simulation import lmoma

In [12]: optgene = OptGene(model)

In [13]: from cameo.flux_analysis import lmoma

In [14]: results = optgene.run(target=model.reactions.EX_vnl_b_glu_c,
                               biomass=model.reactions.BIOMASS_SC5_notrace,
                               substrate=model.reactions.EX_glc__D_e,
                               simulation_method=lmoma)

Starting optimization at Wed, 19 Jul 2017 13:19:12


Finished after 00:37:16


In [15]: results

Out[15]: <cameo.strain_design.heuristic.evolutionary_based.OptGeneResult at 0x10f31bdd8>
```

All things are poison and nothing is without poison. Only the dose makes that a thing is no poison.

Paracelsus

3

MARSI: Metabolite analogues for rational strain improvement

Summary

The forth chapter describes a new approach to CAD of strains in context where GMOs is not allowed and strains are created using CSI or ALE. Metabolites can be 'knocked-out' by blocking the reactions consuming a given metabolites. This chapter has been submitted to Bioinformatics.

Abstract

Summary: Metabolite analogues mimic the structure of native metabolites and can competitively inhibit their utilization and are commonly used as selection tools for isolating desirable mutants of industrial microorganisms. Genome-scale metabolic models GEM representing all biochemical reactions in an organism have proved useful for predicting effects of antimetabolites on cellular phenotype. Here, we present the Metabolite Analogues for Rational Strain Improvement (MARSI) framework. MARSI provides a rational approach to classical strain improvement by searching for metabolites as targets instead of genes or reactions. The designs found by MARSI can be implemented by supplying metabolite analogues in the culture media, which enables metabolic rewiring without the use of recombinant DNA technologies. To facilitate experimental implementation, MARSI provides tools to compare the identified metabolite targets to a database of known drugs and analogues.

Availability and Implementation: The code is freely available at `https://github.com/biosustain/marsi` under the Apache License V2. MARSI is implemented in Python.

Contact: herrgard@biosustain.dtu.dk

Supplementary information: Supplementary data are available at Bioinformatics online.

Introduction

Genome-scale metabolic models GEMs describe the biochemical reactions in an organism and their relation to the proteome and genome (McCloskey et al., 2013). These models comprehensively represent natural metabolism and they are useful for predicting the effect of antimetabolites (Agren et al., 2014). Classical Strain Improvement CSI and adaptive laboratory evolution (ALE) can be used to exploit the evolutionary capacity of microorganisms to improve phenotypes. CSI can be defined as mutagenesis followed by screening (Derkx et al., 2014). ALE is the process of selecting the fittest (e.g., strains that grow faster) under adverse conditions (Hansen et al., 2017). Metabolite analogues, as stress inducers, can be used as the selective pressure in CSI or ALE settings. Here, we present software that implements workflows to identify metabolites as targets instead of gene or reaction knockouts. We also provide a pipeline to identify structural analogues for those targets.

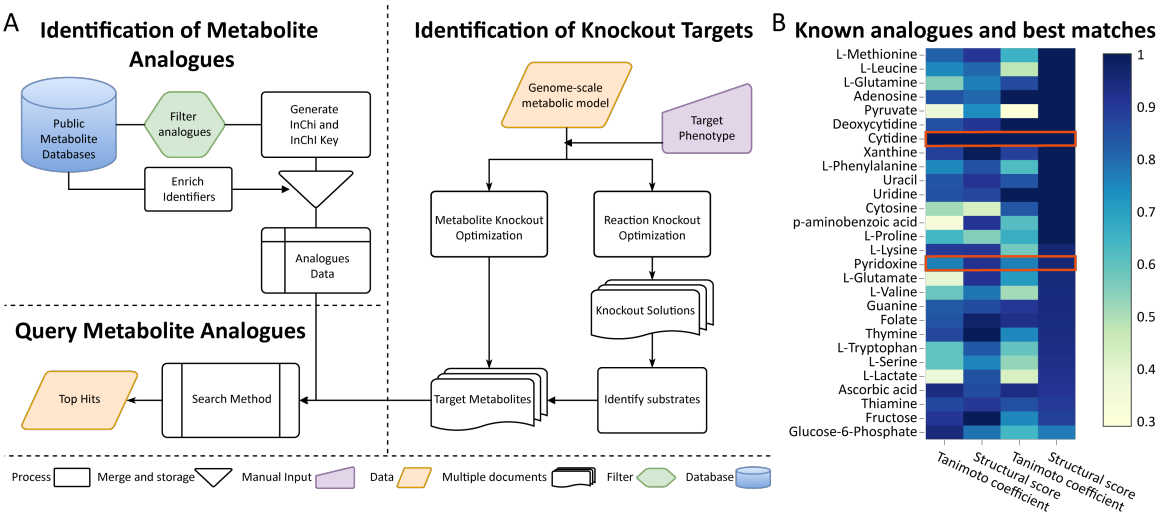


Figure 3.1: Metabolite target identification workflow and examples of metabolite analogues targets. (A) The workflow for obtaining metabolite targets for CSI and ALE. (B) Comparison between the known metabolite analogues (columns 1 and 2) and the best MARSI hits (columns 3 and 4) used to calibrate the search parameters. We show the Tanimoto coefficient and the structural score. We highlighted rows where the best MARSI hit and the known metabolite analogue are the same.

Materials and Methods

The first workflow consists of systematically replacing reaction knockouts (identified a priori by existing strain design methods) by metabolite knockouts until we can find metabolite targets that result in a similar flux distribution. The second workflow consists of searching directly for metabolite targets using heuristic optimization, without the need to specify reaction knockouts a priori. A metabolite knockout consists of blocking all reactions, excluding transporters, consuming a given metabolite. After identifying the metabolite targets, we search for metabolite analogues that could be used to replace them. We compiled a database of potential metabolite analogues from publicly available sources (see 3). We use OpenBabel (O'Boyle et al., 2011) and RDKit (`http://www.rdkit.org`) to calculate the following properties that allow comparing candidate analogues to the metabolite target: number of atoms, number of bonds, number of rings, MACCS fingerprints, Tanimoto coefficient (TC) and structural score (SS).

Results

We implemented a software package containing algorithms to generate strain design strategies using metabolite analogs. Our software could generate metabolite targets for a published knockout-based design (Harder et al., 2016). We also provide the tools to identify candidate metabolite analogues that could be used for implementation of the designs.

Identification of replacement targets

We used an experimentally validated strain design for itaconic acid production in *Escherichia coli* (Harder et al., 2016) and the latest GEM available for *E. coli* iJO1366 (Orth et al., 2014) to demonstrate the use of MARSI. MARSIidentified acetyl-phosphate as a metabolite knockout target that can replace the PTAr reaction knockout and sustain the same flux for itaconic acid production. We use Biomass Product Coupled Yield (Patil et al., 2005) as fitness measure (Table 3.1). Using a SS cutoff of 0.5 (see Supplementary Information 3), we found 182 for Acetyl-Phosphate (Table S3.1 shows the top 10 hits). More examples of replacement targets in other *E. coli* strain designs can be found in Supplementary Information.

Table 3.1: Knockout replacements for the strain design.

Non-replaced knockouts	Replaced reaction	Metabolite	Original fitness	New fitness
PTA2, ICL, ALDD2x, PYK, SUCOAS, GGGABADr	PTAr	Acetyl-P	0.001	0.001

Query calibration with known metabolite analogues

In order to validate the ability of MARSI to find known chemical analogues for a target metabolite, we selected 42 known metabolite-antimetabolite pairs from the literature (Table S3.3). We compared the structural features between the analogues and their target metabolites (Figure S3.3). We used a distance of 4 for the number of atoms, 3 for the number of bonds and 2 for the number of rings as our query cutoff. The TC cutoff changes dynamically with the size of the metabolites

(see 3). In Figure 3.1B, we show the SS and the TC between the targets and their known analogues and the targets and the best hit analogue in the database. MARSI found better or as good candidate analogues as the known examples for most metabolites.

Acknowledgments

We would like to thank Miguel Campodonico for input on chemoinformatics tools.

Funding

This work has been supported by the Novo Nordisk Foundation. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 686070.
Conflict of interest: None declared.

References

Rasmus Agren, A. Mardinoglu, A. Asplund, C. Kampf, M. Uhlen, and Jens Nielsen. Identification of anticancer drugs for hepatocellular carcinoma through personalized genome-scale metabolic modeling. *Molecular Systems Biology*, 10(3):721–721, mar 2014. ISSN 1744-4292. doi: 10.1002/msb.145122.

Patrick M F Derkx, Thomas Janzen, Kim I Sørensen, Jeffrey E Christensen, Birgitte Stuer-Lauridsen, and Eric Johansen. The art of strain improvement of industrial lactic acid bacteria without the use of recombinant DNA technology. *Microbial cell factories*, 13(Suppl 1):S5, 2014. doi: 10.1186/1475-2859-13-S1-S5.

Anne Sofie Lærke Hansen, Rebecca M. Lennen, Nikolaus Sonnenschein, and Markus J Herrgård. Systems biology solutions for biochemical production challenges. *Current opinion in biotechnology*, 45:85–91, jun 2017. ISSN 1879-0429. doi: 10.1016/j.copbio.2016.11.018.

Björn-Johannes Harder, Katja Bettenbrock, and Steffen Klamt. Model-based metabolic engineering enables high yield itaconic acid production by Escherichia coli. *Metabolic Engineering*, 38: 29–37, nov 2016. ISSN 10967176. doi: 10.1016/j.ymben.2016.05.008.

Douglas McCloskey, Bernhard Ø Palsson, and Adam M Feist. Basic and applied uses of genome-scale metabolic network reconstructions of Escherichia coli. *Molecular Systems Biology*, 9(661): 661, jan 2013. ISSN 1744-4292. doi: 10.1038/msb.2013.18.

Noel M. O'Boyle, Michael Banck, Craig A. James, Chris Morley, Tim Vandermeersch, and Geoffrey R. Hutchison. Open Babel: An Open chemical toolbox. *Journal of Cheminformatics*, 3 (10):1–14, 2011. ISSN 17582946. doi: 10.1186/1758-2946-3-33.

Jeffrey D Orth, Tom M Conrad, Jessica Na, Joshua A Lerman, Hojung Nam, Adam M Feist, and Bernhard O. Palsson. A comprehensive genome-scale reconstruction of Escherichia coli

metabolism–2011. *Molecular Systems Biology*, 7(1):535–535, 2014. ISSN 1744-4292. doi: 10.1038/msb.2011.65.

Kiran Raosaheb Patil, Isabel Rocha, Jochen Förster, and Jens Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC bioinformatics*, 6:308, jan 2005. ISSN 1471-2105. doi: 10.1186/1471-2105-6-308.

Supplementary Information

Metabolite knockouts

To knockout a metabolite we block all reactions consuming a given metabolite in all compartments. Because we assume the use of metabolite analogues, we ignore transport reactions. We use a reference flux distribution to identify which reactions consume a metabolite. If no reference is available, we block all reactions consuming that metabolite.

Antimetabolites database

We compiled a comprehensive chemical compound database using publicly available data (see Table S3.1). We also included some known analogues retrieved from the literature (Table S3.2).

Structural similarity

Given two compounds X and Y, we can determine the maximum common substructure (MCS) Z using the RDKit API. Let Z be , the similarity between X and Y is given by the function S as follows:

$$S(X, Y, \alpha, \beta) = s(X, Z, \alpha, \beta) \cdot s(Y, Z, \alpha, \beta)$$

where s is the similarity between a molecule and Z. The function s is defined the following way:

$$\alpha \cdot \frac{Z.atoms}{mol.atoms} + \beta \cdot \frac{Z.bonds}{mol.bonds}$$

The atoms and bonds in Z is always lower or equal to either molecule X or Y. If $alpha + beta = 1$, the similarity (S) is always [0, 1] The MCS algorithm provided by RDKit needs some parameterization. Because we want things that have a similar structure regardless of the atom substitutions we run the algorithm using the following parameters: maximize bounds, ignore atoms types and bond types but still forcing the rings to match.

Dynamic Tanimoto coefficient cutoff

Small metabolites share a lower Tanimoto coefficient with their known analogs (Figure S3.1). To ensure that we can also reach those metabolites, we use a dynamic coefficient based on the following equation:

$$T_{cutoff} = min(0.75, 0.017974 \cdot mol.atoms + 0.008239)$$

The slope of the line was calculated using the linear least-squares regression for between the number of atoms, $mol.atoms$, and the Tanimoto coefficients for all known metabolite analogues. We adjusted the intercept to include all known analogues by subtracting 0.4 to the computed value.

Database Query

The database query is done in three steps: filter metabolites by size (i.e., number of atoms, number of rings and number of bonds), filter metabolites using Tanimoto coefficient, and sort by structural score. In the first step, we retrieve all entries (y) from the database that match the query (x) using the following criteria:

$$x.atoms - 4 \leq y.atoms \leq x.atoms + 4$$
$$x.bonds - 3 \leq y.bonds \leq x.bonds + 3$$
$$x.rings - 2 \leq y.rings \leq x.rings + 2$$

Then we compute the Tanimoto coefficient for all the retrieved entries and we keep the ones above the cutoff. Finally, we compute the structural score for the remaining entries and sort them in descending order using the structural score.

Heuristic optimization search (OptMet)

We implemented a search method that identifies metabolite targets without the need to predict reaction knockouts. To do that, we use evolutionary algorithms. These algorithms use heuristic

approaches inspired in Darwinian evolution to optimize problems with high combinatorial complexity. They do not always guarantee the optimum result, but the solutions found are very close to the global optimum and they have a much lower computational cost. In a simple evolutionary algorithm, there is a representation of the possible states, called the genome (Patil et al., 2005). The solutions as represented as a genome. The genome vector, G, has n elements and the value of each element is True or False. In this case, the n is the number of non-essential metabolites and if the nth position of the vector G is True, then we knockout that metabolite. We store the best results every iteration.

Additional replacement designs

We selected a subset of published growth-coupled designs for Escherichia coli that can be reproduced using the latest E. coli GEM (Table S3.5) (King et al., 2017). Then, we used our algorithm to identify for metabolite knockout targets that could be used to replace one reaction knockout for each design (Table S6). We identified 11 different metabolites that can be used as replacements targets. Computational implementation of the original designs and fitness calculations were done using cameo, a python library for computer-aided metabolic engineering and optimization (Cardoso et al., 2017).

References

João G R Cardoso, Kristian Jensen, Christian Lieven, Anne Sofie Lærke Hansen, Svetlana Galkina, Moritz Beber, Emre Ozdemir, Markus J. Herrgård, Henning Redestig, and Nikolaus Sonnenschein. Cameo : A Python Library for Computer Aided Metabolic Engineering and Optimization of Cell Factories. *bioRxiv*, 9:2013–2018, 2017. doi: 10.1101/147199.

Paula de Matos, Rafael Alcántara, Adriano Dekker, Marcus Ennis, Janna Hastings, Kenneth Haug, Inmaculada Spiteri, Steve Turner, Christoph Steinbeck, and Paula de Matos. Chemical Entities of Biological Interest: an update. *Nucleic Acids Research*, 38(Database issue): D249—-54, jan 2010. ISSN 1362-4962. doi: 10.1093/nar/gkp886.

John J. Irwin, Teague Sterling, Michael M. Mysinger, Erin S. Bolstad, and Ryan G. Coleman. ZINC: A free tool to discover chemistry for biology. *Journal of Chemical Information and Modeling*, 52(7):1757–1768, 2012. ISSN 15499596. doi: 10.1021/ci3001277.

Minoru Kanehisa, Miho Furumichi, Mao Tanabe, Yoko Sato, and Morishima Kanae. KEGG: new perspectives on genomes, pathways, diseases and drugs. *Nucleic Acids Research*, 45 (Database Issue):353–61, 2017. doi: 10.1093/nar/gkw1092.

Sunghwan Kim, Paul A. Thiessen, Evan E. Bolton, Jie Chen, Gang Fu, Asta Gindulyte, Lianyi Han, Jane He, Siqian He, Benjamin A. Shoemaker, Jiyao Wang, Bo Yu, Jian Zhang, and Stephen H. Bryant. PubChem substance and compound databases. *Nucleic Acids Research*, 44(D1):D1202–D1213, 2016. ISSN 13624962. doi: 10.1093/nar/gkv951.

Zachary A King, Edward J. O'Brien, Adam M Feist, and Bernhard O Palsson. Literature mining supports a next-generation modeling approach to predict cellular byproduct secretion. *Metabolic Engineering*, 39(August):220–227, jan 2017. ISSN 10967176. doi: 10.1016/j.ymben.2016.12.004.

Kiran Raosaheb Patil, Isabel Rocha, Jochen Förster, and Jens Nielsen. Evolutionary programming as a platform for in silico metabolic engineering. *BMC bioinformatics*, 6:308, jan 2005. ISSN 1471-2105. doi: 10.1186/1471-2105-6-308.

David S Wishart, C Knox, An C Guo, S Shrivastava, M Hassanali, P Stothard, Z Chang, and J Woolsey. DrugBank: a comprehensive resource for in silico drug discovery and exploration. *Nucleic Acids Research*, 34(90001):D668–D672, 2006. ISSN 0305-1048. doi: 10.1093/nar/gkj067.

Table S3.1: Top 10 analog matches for acetyl-phosphate.

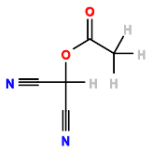
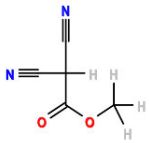
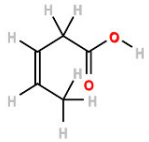
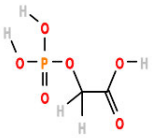
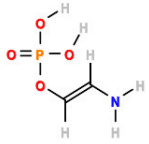
InChI Key	Tanimoto coefficient	Structural similarity	
NDYMFSJJUOHLFJ-UHFFFAOYSA-N	0.28	1.0	
QYIAMKIMEWJTFH-UHFFFAOYSA-N	0.28	0.846	
UIUWNILCHFBLEQ-IHWYPQMZSA-M	0.25	0.783	
ASCFNMCAHFUBCO-UHFFFAOYSA-N	0.567	0.783	
KYMLMTPYCDIFEC-OWOJBTEDSA-N	0.467	0.783	

Table S3.1 – continued from previous

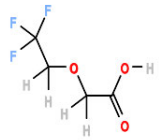
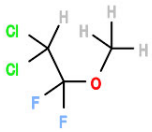
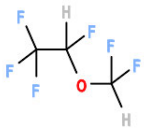
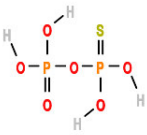
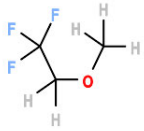
InChI Key	Tanimoto coefficient	Structural similarity	
ICSOIMDWVVEKBW-UHFFFAOYSA-M	0.3	0.783	
RFKMCNOHBTXSMU-UHFFFAOYSA-N	0.25	0.767	
DPYMFVXJLLWWEU-UHFFFAOYSA-N	0.227	0.767	
HWTUHTNZLQJJEV-UHFFFAOYSA-M	0.536	0.767	
OHLVGBXMHDWRRX-UHFFFAOYSA-N	0.214	0.767	

Table S3.2: Chemical Databases and queries used to retrieve the chemical compounds found in the analogues database.

Database	Query/File	Comments
KEGG (Kanehisa et al., 2017)	BR08310	Enzyme inhibitors
ChEBI (de Matos et al., 2010)	CHEBI:35221	'has role' or 'is a' ontology children
PubChem Compound (Kim et al., 2016)	(antimetabolites) OR (analog) OR (analogue)	Retrieved the summary as file
DrugBank (Wishart et al., 2006)	All open structures	
ZINC (Irwin et al., 2012)	'All clean' dataset	From ZINC12

Table S3.3: List of known metabolite-analogue pairs manually collected from databases and literature.

Database	*Identifier*	*Name*	*Target*	*Target Identifier (ChEBI)*	*PMID*	*Group*
PubChem	89034	Methionine sulphoximine	L-Glutamate	29985		Amino acids
ChEBI	74545	acivicin	L-Glutamine	18050		Amino acids
ChEBI	85005	anticapsin	L-Glutamine	18050		Amino acids
ChEBI	74846	azaserine	L-Glutamine	18050		Amino acids
ChEBI	18347	L-norleucine	L-Leucine	15603		Amino acids
ChEBI	42101	D-norleucine	L-Leucine	15603		Amino acids
PubChem	3032849	5-hydroxylysine	L-Lysine	18019	6806159	Amino acids
ChEBI	497734	thialysine	L-Lysine	18019	6806159	Amino acids
PubChem	99558	S-2-aminoethyl-L- cysteine	L-Lysine	18019	1841850	Amino acids
ChEBI	4886	L-ethionine	L-Methionine	16643		Amino acids
PubChem	146719	Thienylalanine	L-Phenylalanine	17295		Amino acids
PubChem	16486	Azetidine carboxylic acid	L-Proline	17203		Amino acids

Table S3.3 – continued from previous page

Database	*Identifier*	*Name*	*Target*	*Target Identifier (ChEBI)*	*PMID*	*Group*
ChEBI	75494	serine hydroxamate	L-Serine	17115	PMC248741	Amino acids
ChEBI	72341	O-(2-aminoethyl)-L-serine	L-Serine	17115		Amino acids
PubChem	150990	5-Methyl-DL-tryptophan	L-Tryptophan	16828		Amino acids
PubChem	9577	DL-5-Fluorotryptophan	L-Tryptophan	16828	PMC429912	Amino acids
ChEBI	18314	L-Norvaline	L-Valine	16414		Amino acids
ChEBI	28804	D-Norvaline	L-Valine	16414		Amino acids
PubChem	5790	Floxuridine	Uridine	16704	PMC2827868	Nucleotides
PubChem	119182	Clofarabine	Adenosine	16335	PMC2827868	Nucleotides
PubChem	439693	Pentostatin	Adenosine	16335	PMC2827868	Nucleotides
PubChem	3011155	Nelarabine	Adenosine	16335	PMC2827868	Nucleotides
PubChem	20279	Cladribine	Adenosine	16335	PMC2827868	Nucleotides
PubChem	451668	Decitabine	Cytidine	17562	PMC2827868	Nucleotides
PubChem	20279	Cladribine	Cytidine	17562	PMC2827868	Nucleotides
PubChem	6253	Cytarabine	Cytidine	17562	PMC2827868	Nucleotides
PubChem	9444	Vidaza	Cytosine	16040	PMC2827868	Nucleotides
PubChem	60750	Gemcitabine	Deoxycytidine	15698	PMC2827868	Nucleotides
PubChem	119182	Clofarabine	Guanine	16235	PMC2827868	Nucleotides
PubChem	2723601	6-thioguanine	Guanine	16235	PMC2827868	Nucleotides

Table S3.3 – continued from previous page

Database	*Identifier*	*Name*	*Target*	*Target Identifier (ChEBI)*	*PMID*	*Group*
PubChem	9444	Vidaza	Thymine	17821	PMC2827868	Nucleotides
PubChem	5802	Bromouracil	Thymine	17821		Nucleotides
PubChem	3385	5-fluorouracil	Uracil	17568	PMC2827868	Nucleotides
ChEBI	28315	alloxanthine	Xanthine	17712		Nucleotides
PubChem	8646	8-Azaguanine	Guanine	16235		Nucleotides
PubChem	974	Oxamate	L-Lactate	16651		Organic Acids
ChEBI	45373	sulfanilamide	p-aminobenzoic acid	30753	PMC3361698	Organic Acids
PubChem	656481	methyl-acetylphosphonate	Pyruvate	15361		Organic Acids
PubChem	73544	2,5-Anhydro-D-mannito	Fructose	28645	8374733	Sugars
PubChem	44099z	2-Deoxy-Glucose-6P	Glucose-6-Phosphate	17665		Sugars
PubChem	54679283	glucoascorbic acid	Ascorbic acid	29073		Vitamins
PubChem	80058	triethylcholine	Choline	15354		Vitamins
PubChem	169371	Aminopterin	Folate	62501		Vitamins
PubChem	6094	4-deoxypyridoxine	Pyridoxine	16709		Vitamins
ChEBI	72290	pyrithiamine	Thiamine	18385		Vitamins

Table S3.4: Strain designs validated experimentally and reproduced *in silico* (King et al., 2017). If the "fva min" value is greater than 0, then the design is growth coupled. "::" means that a heterologous reaction was inserted and "Δ" means that a reaction was knocked out. The "target" refers to exchange reaction for the target product that is produced in the design. The "substrate" column refers to the media composition (assuming a base M9 minimal medium).

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
A1	EX_etoh_e	glucose+LB	anaerobic	0.142	4.8695	::PDC
B1	EX_lac__D_e	glucose	anaerobic	0.1948	16.8641	ΔACALD, ΔALCD2x, ΔPTA2, ΔPTAr
C1	EX_etoh_e	glucose+LB	anaerobic	0.105	0	ΔFRD2, ΔFRD3, ΔOBTFL, ::PDC, ΔPFL
D1	EX_succ_e	glucose+LB	microaerobic	0.1051	0.0329	ΔLDH_D, ΔOBTFL, ΔPFL
E1	EX_succ_e	glucose	anaerobic	0.1892	0.0627	ΔLDH_D, ΔOBTFL, ΔPFL
F1	EX_succ_e	glucose	anaerobic	0.1611	0.0534	ΔACGAptspp, ΔGLCptspp, ΔLDH_D, ΔOBTFL, ΔPFL
G1	EX_lac__L_e	glucose	anaerobic	0.1972	0	ΔLDH_D, ::LDH_L, ΔPTA2, ΔPTAr
H1	EX_lac__D_e	glucose	anaerobic	0.1388	0	ΔPPC, ΔPTA2, ΔPTAr
I1	EX_lac__L_e	xylose	anaerobic	0.1237	0	ΔLDH_D, ::LDH_L, ΔOBTFL, ΔPFL
A2	EX_lac__L_e	glucose	anaerobic	0.1892	0	ΔLDH_D, ::LDH_L, ΔOBTFL, ΔPFL
B2	EX_succ_e	glucose+LB	anaerobic	0.0769	0.0255	ΔACGAptspp, ΔGLCptspp, ΔLDH_D, ΔOBTFL, ΔPFL, ::PYC

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
C2	EX_pyr_e	glucose+acetate	microaerobic	0.1063	0	ΔLDH_D, ΔOBTFL, ΔPDH, ΔPFL, ΔPOX, ΔPPS
D2	EX_lac__L_e	glucose	anaerobic	0.1878	16.7322	ΔACALD, ΔACKr, ΔALCD2x, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔLDH_D, ::LDH_L, ΔOBTFL, ΔPFL
E2	EX_lac__D_e	glucose	anaerobic	0.1878	16.7322	ΔACALD, ΔACKr, ΔALCD2x, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔOBTFL, ΔPFL
F2	EX_ac_e	glucose	aerobic	0.4025	12.7299	ΔACALD, ΔAKGDH, ΔALCD2x, ΔATPS4rpp, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL
G2	EX_pyr_e	glucose+acetate	microaerobic	0.1063	0	ΔLDH_D, ΔOBTFL, ΔPDH, ΔPFL, ΔPOX, ΔPPS
H2	EX_lac__D_e	glucose	anaerobic	0.1948	16.8641	ΔACALD, ΔALCD2x, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
I2	EX_lac__D_e	glucose	anaerobic	0.193	0	ΔPFK, ΔPFK_2, ΔPFK_3, ΔPTA2, ΔPTAr
A3	EX_lac__D_e	glucose	anaerobic	0.1908	16.929	ΔACALD, ΔALCD2x, ΔHEX1, ΔPFK, ΔPFK_2, ΔPFK_3, ΔPTA2, ΔPTAr
B3	EX_succ_e	glucose+LB	microaerobic	0	0	ΔACKr, ΔICDHyr, ΔPOX, ΔPTA2, ΔPTAr, ΔSUCDi
C3	EX_succ_e	glucose+LB	anaerobic	0.085	0.0282	ΔACALD, ΔACGAptspp, ΔALCD2x, ΔGLCptspp, ΔLDH_D, ::PYC
D3	EX_succ_e	pyruvate+yeast extract	anaerobic	0	0	ΔACGAptspp, ΔGLCptspp, ΔPYK
E3	EX_succ_e	glucose+LB	microaerobic	0	0	ΔACKr, ΔICDHyr, ΔPOX, ΔPTA2, ΔPTAr, ΔSUCDi
F3	EX_succ_e	glucose	anaerobic	0.1639	0.0543	ΔACALD, ΔACKr, ΔALCD2x, ΔLDH_D, ΔPTA2, ΔPTAr, ::PYC
G3	EX_succ_e	glucose+yeast extract	anaerobic	0.0674	0.0223	ΔACGAptspp, ΔGLCptspp, ΔPYK

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
H3	EX_lac__D_e	glucose	anaerobic	0.1878	16.7322	ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔOBTFL, ΔPFL
I3	EX_lac__L_e	glucose	anaerobic	0.1872	17.2334	ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ::LDH_L, ΔMGSA, ΔOBTFL, ΔPFL
A4	EX_lac__D_e	glucose+betain	anaerobic	0.1878	16.7322	ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔOBTFL, ΔPFL
B4	EX_lac__D_e	glucose	anaerobic	0.1872	17.2334	ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔMGSA, ΔOBTFL, ΔPFL
C4	EX_ala__L_e	glucose+betain	anaerobic	0.0829	0	ΔACALD, ΔACKr, ::ALADH_L, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
D4	EX_ala__L_e	glucose+bet:	anaerobic	0	0	ΔACALD, ΔACKr, ::ALADH_L, ΔALAR, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL
E4	EX_h2_e	glucose	anaerobic	0.2878	16.7623	ΔFDH4pp, ΔFDH5pp, ΔFRD2, ΔFRD3, ΔHYD1pp, ΔHYD2pp, ΔHYD3pp, ΔLDH_D, ::PDC, ΔPDH
F4	EX_lac__D_e	glucose	anaerobic	0.1889	17.2075	ΔFRD2, ΔFRD3, ΔOBTFL, ΔPDH, ΔPFL, ΔPOX, ΔPPS
G4	EX_etoh_e	glycerol	anaerobic	0.0541	0.7119	ΔFRD2, ΔFRD3, ΔPTA2, ΔPTAr
H4	EX_1poh_e	D-Glucose+L-Valine+L-Isoleucine+I Leucine	microaerobic	0	0	::1PDH, ::2OBUTDC, ΔACALD, ΔALCD2x, ΔDHAD1, ΔDHAD2, ::EX_1poh_e, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
I4	EX_iboh_e	glucose	microaerobic	0.164	0	::3MOBDC, ΔACALD, ΔALCD2x, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ::EX_iboh_e, ΔFRD2, ΔFRD3, ::IBDH, ΔLDH_D, ΔPTA2, ΔPTAr
A5	EX_1boh_e	D-Glucose+L-Valine+L-Isoleucine+I Leucine	microaerobic	0	0	::1BDH, ::2KVDC, ΔACALD, ΔALCD2x, ΔDHAD1, ΔDHAD2, ::EX_1boh_e, ::EX_2phetoh_e, ΔFRD2, ΔFRD3, ::ILV_PATHWAY, ΔLDH_D, ΔPTA2, ΔPTAr
B5	EX_etoh_e	glycerol	anaerobic	0.0541	0.7119	ΔFHL, ΔFRD2, ΔFRD3, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
C5	EX_etoh_e	glucose+LB	anaerobic	0.0792	5.3695	ΔACGAptspp, ΔACMANAptspp, ΔFRD2, ΔFRD3, ΔFRUpts2pp, ΔG6PDH2r, ΔGAMptspp, ΔGLCptspp, ΔHEX1, ΔLDH_D, ΔMANptspp, ΔME2, ΔNADH10, ΔNADH5, ΔNADH9, ΔPOX, ΔPTA2, ΔPTAr
D5	EX_iamoh_e	glucose	anaerobic	0	0	::3M1BDH, ::4MOBDC, ΔACALD, ΔALCD2x, ::EX_iamoh_e, ΔFRD2, ΔFRD3, ΔILETA, ΔLDH_D, ΔLEUTAi, ΔOBTFL, ΔPFL, ΔPHETA1, ΔPTA2, ΔPTAr, ΔTYRTA, ΔVALTA

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
E5	EX_succ_e	glucose	anaerobic	0	0	ΔACALD, ΔACKr, ΔALCD2x, ΔASPTA, ΔCITL, ΔFORt2pp, ΔFORtppi, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔPHETA1, ΔPOX, ΔPTA2, ΔPTAr, ΔTYRTA
F5	EX_1boh_e	glucose	microaerobic	0.1972	8.2255	::1BDH, ΔACALD, ΔALCD2x, ::B2COAR, ::BTALDH, ::EX_1boh_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ΔPTAr
G5	EX_succ_e	glucose	anaerobic	0	0	ΔACALD, ΔACKr, ΔALCD2x, ΔASPTA, ΔCITL, ΔFORt2pp, ΔFORtppi, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔPHETA1, ΔPOX, ΔPTA2, ΔPTAr, ΔTYRTA
H5	EX_lac__D_e	glucose	aerobic	0.8471	0	ΔCYTBDpp, ΔCYTBO3_4pp, ΔQMO2, ΔQMO3

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
I5	EX_succ_e	glucose+beta	anaerobic	0.1081	7.673	ΔACALD, ΔACKr, ΔALCD2x, ΔFORt2pp, ΔFORtppi, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔPOX
A6	EX_btd__RR_e	glucose+yeast extract	microaerobic	0.0502	4.851	ΔACALD, ::ACLDC, ΔALCD2x, ::EX_btd__RR_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ::sADHx, ::sADHy
B6	EX_lac__D_e	glucose	anaerobic	0.1878	16.7322	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔOBTFL, ΔPFL
C6	EX_btd__meso_e	glucose+yeast extract	microaerobic	0.0502	4.851	ΔACALD, ::ACLDC, ΔALCD2x, ::EX_btd__meso_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ::sADHx
D6	EX_succ_e	glycerol	anaerobic	0.0387	0.0128	ΔACALD, ΔALCD2x, ΔLDH_D, ΔPOX, ΔPPC, ΔPTA2, ΔPTAr, ::PYC

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
E6	EX_3hb_e	glucose	anaerobic	0.1639	0	::3HB_POLYM, ΔACALD, ΔACKr, ΔALCD2x, ::EX_3hb_e, ::EX_3hv_e, ΔLDH_D, ::PHPB, ΔPOX, ΔPTA2, ΔPTAr
F6	EX_h2_e	glycerol	anaerobic	0.1264	16.9207	ΔFRD2, ΔFRD3
G6	EX_succ_e	glycerol	anaerobic	0	0	ΔACGAptspp, ΔACMANAptspp, ΔACMUMptspp, ΔARBTptspp, ΔASCBptspp, ΔCHTBSptspp, ΔDHAPT, ΔFRUpts2pp, ΔFRUptspp, ΔGALTptspp, ΔGAMptspp, ΔGLCptspp, ΔMALTptspp, ΔMANGLYCptspp, ΔMANptspp, ΔMNLptspp, ΔOBTFL, ΔPFL, ΔSBTptspp, ΔSUCptspp, ΔTREptspp

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
H6	EX_lac__D_e	glucose	microaerobic	0.1273	10.0258	ΔACALD, ΔACKr, ΔALCD2x, ΔPPC
I6	EX_3hb_e	glucose	anaerobic	0.1972	0	::3HB_POLYM, ΔACKr, ::EX_3hb_e, ::EX_3hv_e, ΔLDH_D, ::PHPB, ΔPOX, ΔPTA2, ΔPTAr
A7	EX_3hb_co_l	glucose	microaerobic	0.1273	0	::3HB_LA_POLYM, ΔACALD, ΔACKr, ΔALCD2x, ::EX_3hb_co_la_e, ::PHPB, ΔPPC
B7	EX_iboh_e	glucose+yeast extract	anaerobic	0.0715	5.8784	::3MOBDC, ΔACALD, ΔALCD2x, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ::EX_iboh_e, ΔFRD2, ΔFRD3, ::IBDH, ΔLDH_D, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr
C7	EX_lac__D_e	glycerol	anaerobic	0.0387	0	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔLDH_D2, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
D7	EX_3hb_e	glucose	anaerobic	0.1081	0	::3HB_POLYM, ΔACALD, ΔACKr, ΔALCD2x, ::EX_3hb_e, ::EX_3hv_e, ΔLDH_D, ΔOBTFL, ΔPFL, ::PHPB, ΔPOX, ΔPTA2, ΔPTAr
E7	EX_14btd_e	glucose	microaerobic	0.1791	5.0499	::4HBACT, ::4HBTALDDH, ΔACALD, ::AKGDC, ΔALCD2x, ::BTDP2, ::EX_14btd_e, ::EX_4hdxbld_e, ::EX_gbl_e, ::GBL_PROD, ΔLDH_D, ΔMDH, ΔOBTFL, ΔPFL, ::SUCCALDH

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
F7	EX_iboh_e	glucose+yeast extract	anaerobic	0.1077	0	::3MOBDC, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ::EX_iboh_e, ΔFRD2, ΔFRD3, ΔG6PDH2r, ::IBDH, ΔLDH_D, ΔMDH, ΔNADH10, ΔNADH5, ΔNADH9, ΔPOX, ΔPTA2, ΔPTAr
G7	EX_1boh_e	glucose+yeas extract	anaerobic	0.1181	4.6922	::1BDH, ΔACALD, ΔALCD2x, ::B2COAR, ::BTALDH, ::EX_1boh_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ΔPTAr
H7	EX_1boh_e	glucose	anaerobic	0.1972	1.2106	::1BDH, ΔACALD, ΔALCD2x, ::B2COAR, ::BTALDH, ::EX_1boh_e, ΔFRD2, ΔFRD3, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
I7	EX_succ_e	LB+glucose, sorbitol, and gluconate	microaerobic	0.0303	8.404	ΔACALD, ΔACGAptspp, ΔALCD2x, ΔGLCptspp, ΔLDH_D, ΔOBTFL, ΔPFL
A8	EX_lac__D_e	glucose	microaerobic	0.1879	16.7322	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔOBTFL, ΔPFL
B8	EX_1hex_e	glucose+yeas extract	anaerobic	0.1181	3.1281	::1HDH, ΔACALD, ΔALCD2x, ::EX_1hex_e, ΔFRD2, ΔFRD3, ::HX2COAR, ::HXALDH, ΔLDH_D, ΔPTA2, ΔPTAr
C8	EX_iboh_e	glucose	anaerobic	0.1388	8.7924	::3MOBDC, ΔACALD, ΔALCD2x, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ::EX_iboh_e, ΔFRD2, ΔFRD3, ::IBDH, ::KARA1x, ΔLDH_D, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
D8	EX_mal__L_e	glucose	anaerobic	0.0798	0	ΔACALD, ΔACKr, ΔALCD2x, ΔDTARTD, ΔFRD2, ΔFRD3, ΔFUM, ΔLDH_D, ΔME2, ΔMGSA, ΔOBTFL, ΔPFL, ΔPOX
E8	EX_lac__D_e	glucose	microaerobic	0.0846	0	ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔLDH_D2, ΔOBTFL, ΔPFL, ΔPOX, ΔPPS, ΔPTA2, ΔPTAr
F8	EX_3hb_co_la	glucose	microaerobic	0.1879	0	::3HB_LA_POLYM, ΔACALD, ΔALCD2x, ::EX_3hb_co_la_e, ΔFRD2, ΔFRD3, ΔOBTFL, ΔPFL, ::PHPB
G8	EX_lac__D_e	glycerol	microaerobic	0	0	ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔLDH_D2, ΔOBTFL, ΔPFL, ΔPOX, ΔPPS, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
H8	EX_etoh_e	glucose	microaerobic	0.1893	17.204	ΔACKr, ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL
I8	EX_etoh_e	xylose	microaerobic	0.1238	14.839	ΔACKr, ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL
A9	None	glucose	microaerobic	0	0	ΔACALD, ΔALCD2x, ::EX_2ptone_e, ΔLDH_D, ::MKS, ΔPOX, ΔPTA2, ΔPTAr, ::THE
B9	EX_1boh_e	glucose+LB	anaerobic	0.1039	5.7324	::1BDH, ΔACALD, ΔACKr, ΔALCD2x, ::B2COAR, ::BTALDH, ::EX_1boh_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL
C9	EX_lac__D_e	sucrose	anaerobic	0	0	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔGCALDD, ΔLCADi, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
D9	EX_3hb_e	glucose	anaerobic	0.2415	0	::3HB_POLYM, ::EX_3hb_e, ΔFDH4pp, ΔFDH5pp, ΔFRD2, ΔFRD3, ΔHYD1pp, ΔHYD2pp, ΔHYD3pp, ΔLDH_D, ΔPDH, ::PHPB
E9	EX_xylt_e	glucose+xylc	anaerobic	0.1259	0	ΔACALD, ΔACGAptspp, ΔACKr, ΔALCD2x, ::EX_xylt_e, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔGLCptspp, ΔLDH_D, ::XYLR
F9	EX_1poh_e	glucose+yeast extract	anaerobic	0	0	::1PDH, ::2OBUTDC, ΔACALD, ΔACHBS, ΔACLS, ΔALCD2x, ::BMALDH, ::BMALHYD, ::CIMHYD, ::CIMSY, ::EX_1poh_e, ΔFRD2, ΔFRD3, ΔIPPS, ΔLDH_D, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
G9	EX_lac__L_e	glycerol	microaerobic	0.0501	7.2256	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ::LDH_L, ΔMGSA, ΔPTA2, ΔPTAr
H9	EX_1boh_e	glucose	anaerobic	0.1972	8.225	::1BDH, ΔACACCT, ΔACALD, ΔALCD2x, ::B2COAR, ::BTALDH, ΔBUTCT, ΔDHACOAH, ::EX_1boh_e, ΔFRD2, ΔFRD3, ΔHADPCOADH3, ΔHXCT, ΔLDH_D, ΔOXDHCOAT, ΔPTA2, ΔPTAr, ΔREPHACCOAI
I9	EX_succ_e	glucose+LB	anaerobic	0.105	0.0348	ΔLDH_D, ΔOBTFL, ΔPFL, ::PYC

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
A10	EX_but_e	glucose	anaerobic	0.1639	0	ΔACACCT, ΔACALD, ΔALCD2x, ::B2COAR, ΔBUTCT, ::BUTTH, ΔDHACOAH, ΔFRD2, ΔFRD3, ΔHADPCOADH3, ΔHXCT, ΔLDH_D, ΔOXDHCOAT, ΔPTA2, ΔPTAr, ΔREPHACCOAI
B10	EX_h2_e	glycerol	anaerobic	0.0868	17.3145	ΔFDH4pp, ΔFDH5pp, ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔNO3R1pp, ΔNO3R2pp, ΔPPC
C10	EX_but_e	glucose	anaerobic	0.1659	0	ΔACALD, ΔACGAptspp, ΔALCD2x, ::B2COAR, ::BUTTH, ΔFRD2, ΔFRD3, ΔGLCptspp, ΔLDH_D, ΔPOX
D10	EX_but_e	glucose+LB	anaerobic	0.0918	0	ΔACALD, ΔACKr, ΔALCD2x, ::B2COAR, ::BUTTH, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
E10	EX_lac__L_e	mannitol	microaerobic	0.1082	0	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ::LDH_L, ΔL_LACD2, ΔL_LACD3, ΔMGSA, ΔPTA2, ΔPTAr
F10	EX_1boh_e	glucose+yeas extract	anaerobic	0	0	::1BDH, ΔACALD, ΔACOLIPAabctex, ΔALCD2x, ::B2COAR, ::BTALDH, ΔCLIPAabctex, ΔCOLIPAPabctex, ΔCOLIPAabctex, ΔECA4COLIPAabctex, ΔENLIPAabctex, ::EX_1boh_e, ΔFRD2, ΔFRD3, ΔK2L4Aabctex, ΔLDH_D, ΔLIPAabctex, ΔO16A4COLIPAabctex, ΔPTA2, ΔPTAr
G10	EX_crot_e	glycerol	anaerobic	0.0387	0	ΔACALD, ΔALCD2x, ::CROT, ::EX_crot_e, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPOX, ΔPTA2, ΔPTAr

Table S3.4 – continued from previous page

Key	*Target*	*Substrate*	*Aerobicity*	*Growth rate*	*Fva min*	*Strain design*
H10	EX_xylt_e	glucose+xylc	anaerobic	0.0538	0	ΔACALD, ΔACGAptspp, ΔACKr, ΔALCD2x, ΔDXYLK, ::EX_xylt_e, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔGLCptspp, ΔLDH_D, ΔXYLI1, ΔXYLI2, ΔXYLK, ::XYLR

Table S3.5: Predicted metabolite targets and correspondent reaction knockout targets identified by MARSI for strain designs that have been experimentally validated and reproduced in silico. For most designs, it is not possible to find an analogues-only design, but rather some of the gene deletions can be replaced by metabolite analogues.

Key	*Base design*	*Replaced target*	*Metabolite target*	*Old Fitness*	*New Fitness*
B1	ΔACALD, ΔALCD2x, ΔPTA2	ΔPTAr	actp	0.3285	0.3285
D2	ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ΔALCD2x, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ::LDH_L	ΔLDH_D	lac-D	0.3143	0.3143
F2	ΔATPS4rpp, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL, ΔACALD, ΔAKGDH	ΔALCD2x	acald, etoh	0.5123	0.5124
F2	ΔLDH_D, ΔOBTFL, ΔPFL, ΔACALD, ΔAKGDH, ΔALCD2x, ΔATPS4rpp, ΔFORt2pp, ΔFORtppi, ΔFRD2	ΔFRD3	2dmmq8	0.5123	0.5124
F2	ΔPFL, ΔACALD, ΔAKGDH, ΔALCD2x, ΔATPS4rpp, ΔFORt2pp, ΔFORtppi, ΔFRD2, ΔFRD3, ΔLDH_D	ΔOBTFL	ppcoa	0.5123	0.5124
H2	ΔPTA2, ΔPTAr, ΔACALD	ΔALCD2x	acald, etoh	0.3285	0.3285
H2	ΔACALD, ΔALCD2x, ΔPTA2	ΔPTAr	actp	0.3285	0.3285
A3	ΔACALD, ΔALCD2x, ΔHEX1, ΔPFK, ΔPFK_2, ΔPFK_3, ΔPTA2	ΔPTAr	actp	0.323	0.323

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
I3	ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ::LDH_L	ΔACKr	actp	0.3225	0.3225
I3	ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ::LDH_L	ΔALCD2x	acald	0.3225	0.319
I3	ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ::LDH_L	ΔALCD2x	etoh	0.3225	0.3225
I3	ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ::LDH_L	ΔLDH_D	lac-D	0.3225	0.3225
I3	ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ::LDH_L	ΔMGSA	mthgxl	0.3225	0.3225
B4	ΔALCD2x, ΔFRD2, ΔFRD3, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD	ΔACKr	actp	0.3225	0.3225
B4	ΔFRD2, ΔFRD3, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr	ΔALCD2x	acald	0.3225	0.319
B4	ΔFRD2, ΔFRD3, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr	ΔALCD2x	etoh	0.3225	0.3225

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
B4	ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3	ΔMGSA	mthgxl	0.3225	0.3225
G4	ΔFRD3, ΔPTA2, ΔPTAr	ΔFRD2	succ	0.0039	0.0042
G4	ΔPTA2, ΔPTAr, ΔFRD2	ΔFRD3	succ	0.0039	0.0042
B5	ΔFRD3, ΔPTA2, ΔPTAr, ΔFHL	ΔFRD2	succ	0.0039	0.0042
B5	ΔPTA2, ΔPTAr, ΔFHL, ΔFRD2	ΔFRD3	succ	0.0039	0.0042
C5	ΔMANptspp, ΔME2, ΔNADH10, ΔNADH5, ΔNADH9, ΔPOX, ΔPTA2, ΔPTAr, ΔACGAptspp, ΔACMANAptspp, ΔFRD2, ΔFRD3, ΔFRUpts2pp, ΔG6PDH2r, ΔGAMptspp, ΔGLCptspp, ΔHEX1	ΔLDH_D	lac-D	0.2771	0.2771
C5	ΔACGAptspp, ΔACMANAptspp, ΔFRD2, ΔFRD3, ΔFRUpts2pp, ΔG6PDH2r, ΔGAMptspp, ΔGLCptspp, ΔHEX1, ΔLDH_D, ΔMANptspp, ΔME2, ΔNADH10, ΔNADH5, ΔNADH9, ΔPOX, ΔPTA2	ΔPTAr	actp	0.2771	0.2771

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
F5	ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔLDH_D	lac-D	0.1622	0.1622
F5	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔPTAr	actp	0.1622	0.1622
I5	ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔPOX, ΔACALD, ΔACKr, ΔALCD2x, ΔFORt2pp	ΔFORtppi	for	0.083	0.083
I5	ΔMGSA, ΔOBTFL, ΔPFL, ΔPOX, ΔACALD, ΔACKr, ΔALCD2x, ΔFORt2pp, ΔFORtppi	ΔLDH_D	lac-D	0.083	0.083
A6	ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ΔACALD, ::ACLDC, ::EX_btd__RR_e, ::sADHx, ::sADHy	ΔALCD2x	acald, etoh	0.0806	0.0806
A6	ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::ACLDC, ::EX_btd__RR_e, ::sADHx, ::sADHy	ΔLDH_D	lac-D	0.0806	0.0806

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
C6	ΔFRD2, ΔFRD3, ΔLDH_D, ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ΔACALD, ::ACLDC, ::EX_btd__meso_e, ::sADHx	ΔALCD2x	acald, etoh	0.0806	0.0806
C6	ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::ACLDC, ::EX_btd__meso_e, ::sADHx	ΔLDH_D	lac-D	0.0806	0.0806
H6	ΔALCD2x, ΔPPC, ΔACALD	ΔACKr	ac, actp	0.1277	0.1277
B7	ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::3MOBDC, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ::EX_iboh_e, ::IBDH	ΔLDH_D	lac-D	0.1462	0.1462
E7	ΔMDH, ΔOBTFL, ΔPFL, ΔACALD, ΔALCD2x, ::4HBACT, ::4HBTALDDH, ::AKGDC, ::BTDP2, ::EX_14btd_e, ::EX_4hdxbld_e, ::EX_gbl_e, ::GBL_PROD, ::SUCCALDH	ΔLDH_D	lac-D	0.0904	0.0904
G7	ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔLDH_D	lac-D	0.1725	0.1725

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
G7	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔPTAr	actp	0.1725	0.1725
H7	ΔFRD3, ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔFRD2	succ	0.0239	0.0264
H7	ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔFRD3	succ	0.0239	0.0264
I7	ΔLDH_D, ΔOBTFL, ΔPFL, ΔACALD, ΔACGAptspp, ΔALCD2x	ΔGLCptsp	g6p	0.1197	0.1197
I7	ΔOBTFL, ΔPFL, ΔACALD, ΔACGAptspp, ΔALCD2x, ΔGLCptspp	ΔLDH_D	lac-D	0.1197	0.1197
B8	ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::1HDH, ::EX_1hex_e, ::HX2COAR, ::HXALDH	ΔLDH_D	lac-D	0.115	0.115
B8	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔPTA2, ::1HDH, ::EX_1hex_e, ::HX2COAR, ::HXALDH	ΔPTAr	actp	0.115	0.115

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
C8	ΔOBTFL, ΔPFL, ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::3MOBDC, ::EX_2mbtoh_e, ::EX_2phetoh_e, ::EX_iamoh_e, ::EX_iboh_e, ::IBDH, ::KARA1x	ΔLDH_D	lac-D	0.1221	0.1221
H8	ΔOBTFL, ΔPFL, ΔACKr, ΔFRD2, ΔFRD3	ΔLDH_D	lac-D	0.3256	0.3256
I8	ΔOBTFL, ΔPFL, ΔACKr, ΔFRD2, ΔFRD3	ΔLDH_D	lac-D	0.1837	0.1837
B9	ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔALCD2x	acald	0.1709	0.1693
B9	ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔALCD2x	etoh	0.1709	0.1709
B9	ΔMGSA, ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔLDH_D	lac-D	0.1709	0.1709

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
B9	ΔOBTFL, ΔPFL, ΔACALD, ΔACKr, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔMGSA	mthgxl	0.1709	0.1709
G9	ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔPTA2, ΔPTAr, ΔACALD, ::LDH_L	ΔALCD2x	acald, etoh	0.0362	0.0362
G9	ΔMGSA, ΔPTA2, ΔPTAr, ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ::LDH_L	ΔLDH_D	lac-D	0.0362	0.0362
G9	ΔACALD, ΔALCD2x, ΔFRD2, ΔFRD3, ΔLDH_D, ΔMGSA, ΔPTA2, ::LDH_L	ΔPTAr	actp	0.0362	0.0362
H9	ΔOXDHCOAT, ΔPTA2, ΔPTAr, ΔREPHACCOAI, ΔACACCT, ΔACALD, ΔALCD2x, ΔBUTCT, ΔDHACOAH, ΔFRD2, ΔFRD3, ΔHADPCOADH3, ΔHXCT, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔLDH_D	lac-D	0.1622	0.1622

Table S3.5 – continued from previous page

Key	*Base design*	*Replaced Target*	*Metabolite Target*	*Old Fitness*	*New Fitness*
H9	ΔREPHACCOAI, ΔACACCT, ΔACALD, ΔALCD2x, ΔBUTCT, ΔDHACOAH, ΔFRD2, ΔFRD3, ΔHADPCOADH3, ΔHXCT, ΔLDH_D, ΔOXDHCOAT, ΔPTA2, ::1BDH, ::B2COAR, ::BTALDH, ::EX_1boh_e	ΔPTAr	actp	0.1622	0.1622
B10	ΔLDH_D, ΔMGSA, ΔNO3R1pp, ΔNO3R2pp, ΔPPC, ΔFDH4pp, ΔFDH5pp, ΔFRD2	ΔFRD3	succ	0.1503	0.1421
B10	ΔLDH_D, ΔMGSA, ΔNO3R1pp, ΔNO3R2pp, ΔPPC, ΔFDH4pp, ΔFDH5pp, ΔFRD2	ΔFRD3	2dmmq8	0.1503	0.1503
B10	ΔMGSA, ΔNO3R1pp, ΔNO3R2pp, ΔPPC, ΔFDH4pp, ΔFDH5pp, ΔFRD2, ΔFRD3	ΔLDH_D	lac-D	0.1503	0.1503
B10	ΔNO3R1pp, ΔNO3R2pp, ΔPPC, ΔFDH4pp, ΔFDH5pp, ΔFRD2, ΔFRD3, ΔLDH_D	ΔMGSA	mthgxl	0.1503	0.1503

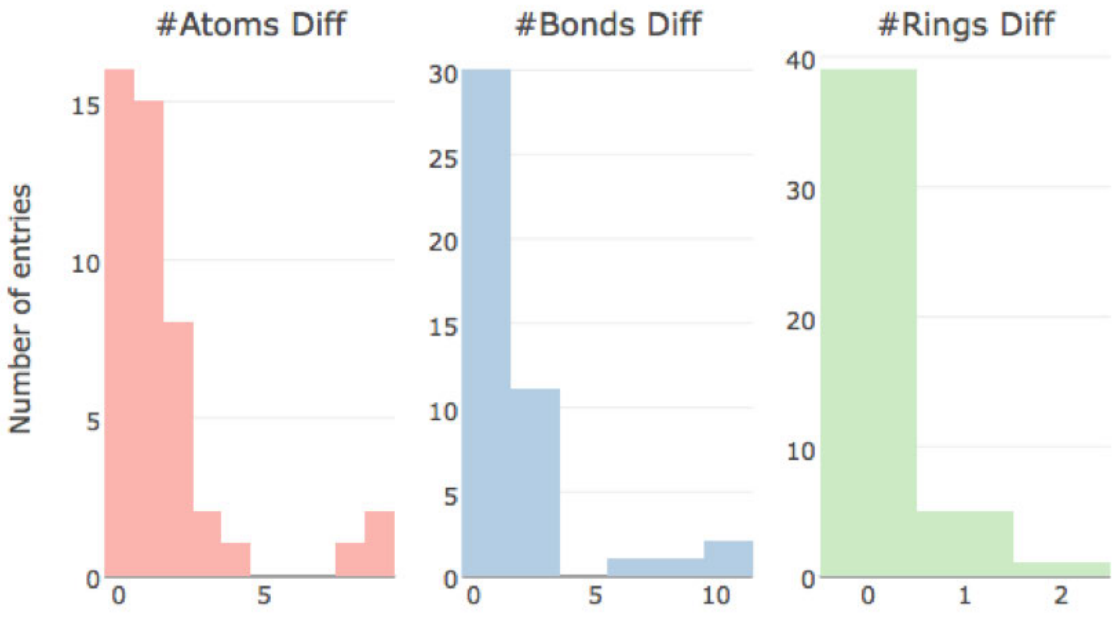


Figure S3.1: Metabolite analogues retrieved using different cutoffs. Number of metabolite analogues from our selected metabolite-antimetabolite pairs that can be retrieved using the difference between the number of atoms, the number of bonds and the number of rings using different cutoffs.

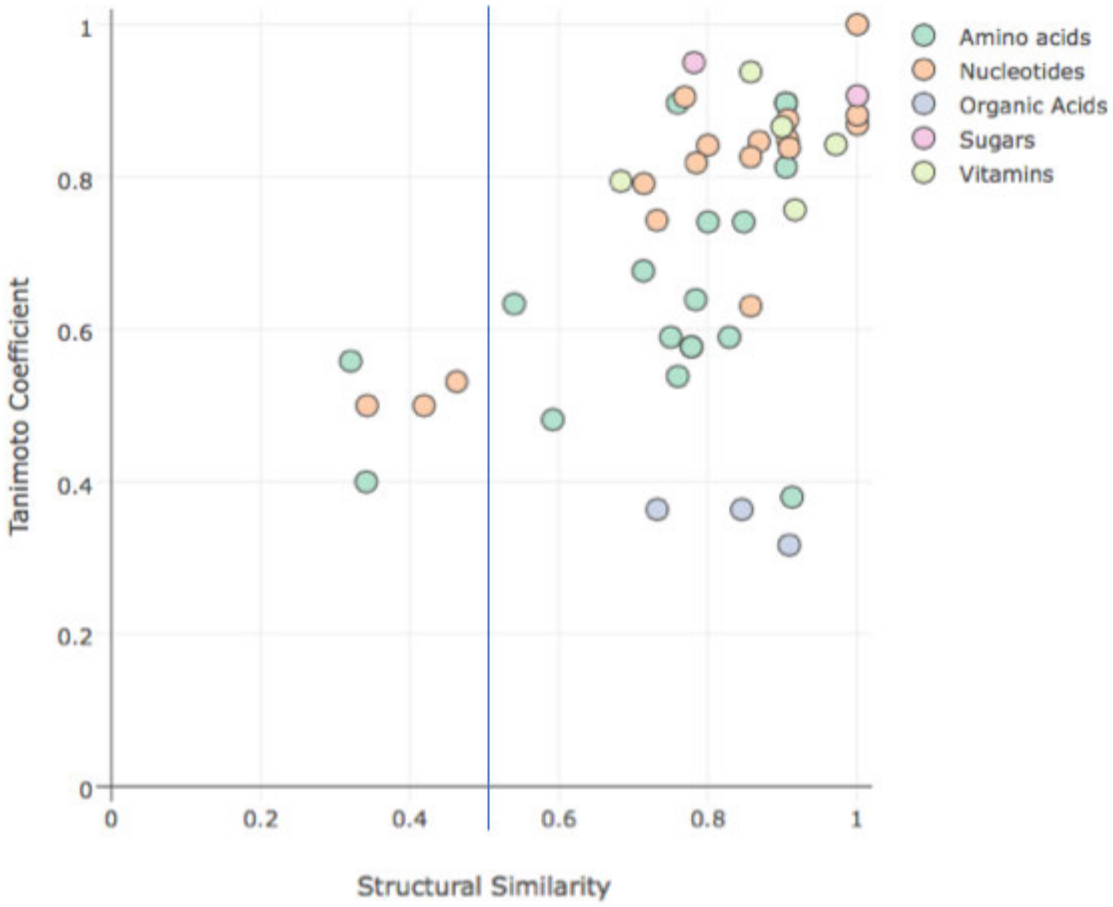


Figure S3.2: Comparison between Tanimoto coefficients and structural similarity. We can capture almost all known metabolite analogues with a similarity of 0.5.

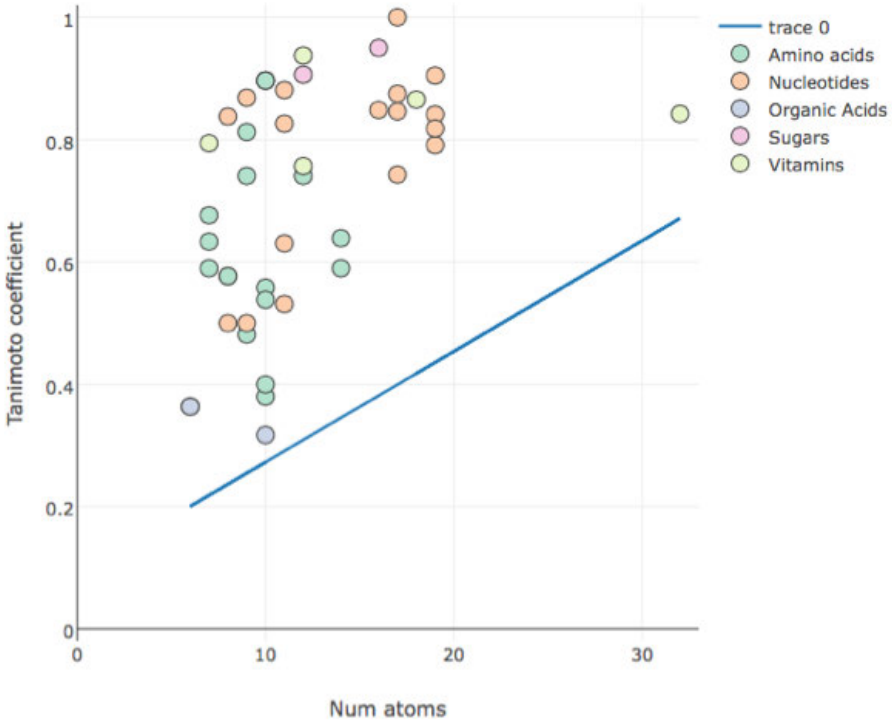


Figure S3.3: Tanimoto coefficient vs. number of atoms. The distribution of the number of atoms per molecule. The line represents the linear regression with adjusted intercept. This line was used to determine the cutoff value for smaller compounds.

He who loves practice without theory is like the sailor who boards ship without a rudder and compass and never knows where he may cast.

Leonardo da Vinci

4

Improving mevalonate production in *Saccharomyces cerevisiae* using constraint based modeling

Summary

In this chapter, the computer-aided design methods implemented in *cameo* were used to design a *S. cerevisiae* platform strain with increased flux through the mevalonate pathway. The strains are being implemented in the laboratory and the results enclosed here are preliminary. Because of the intellectual property value of this work, many details including the designs, genotypes and results from *in silico* simulations cannot be disclosed.

Abstract

Mevalonate is the precursor of a large range of valuable chemicals. *Saccharomyces cerevisiae* has a long history of metabolic engineering applications and contains a native mevalonate pathway. In this work, we engineered *S. cerevisiae* metabolism to increase the flux through the mevalonate pathway. We used computer-aided design methods to search and evaluate different engineering strategies. Using the results obtained *in silico*, we applied genetic modifications (insertion of heterologous genes and knockouts) that can shift the redox metabolism in the cytosol and increase the availability of the cofactors necessary to produce mevalonate. We tested our hypothesis using a NADPH/$NADP^+$ biosensor and the β-Carotene pathway.

Introduction

Mevalonate is an essential precursor involved in cellular processes and leading to large range of valuable chemicals. These include pharmaceutical (e.g., anti-malarial drug artemisinin, phytosterols with antioxidant and anti-cancer activity), fuels (e.g., farnesene), waterproof products and food additives, such as carotenes (Liao et al., 2016, Zhang et al., 2011). Cell factories with enhanced mevalonate pathway provide a great platform strain for production of these valuable chemicals.

Saccharomyces cerevisiae (Bakers' yeast) is the main model organism for yeast species and has a long history for metabolic engineering applications. It is well studied and generally recognized as safe (GRAS) (Nevoigt, 2008). The mevalonate pathway is present and well studied in *S. cerevisiae*, making it our host of choice for this application.

Mevalonate is produced from acetyl-CoA in three steps. In the first step, two acetyl-CoA molecules are combined into acetoacetyl-CoA by an acetoacetyl-CoA thiolase (ERG10). The following conversion is carried by 3-hydroxy-3-methylglutaryl-CoA (HMG-CoA) synthase (ERG13), that converts acetoacetyl-CoA into HMG-CoA. In the last step, HMG-CoA is reduced to mevalonate using two NADPH molecules (Figure 4.1).

Genome-scale metabolic models GEMs are stoichiometric models that describe the portfolio of biochemical reactions in organisms. These models have proven useful for metabolic engineering because they are capable of predicting the phenotype resulting from genetic interventions (i.e., gene knockouts, knock-ins and expression changes) and media composition (O'Brien et al.,

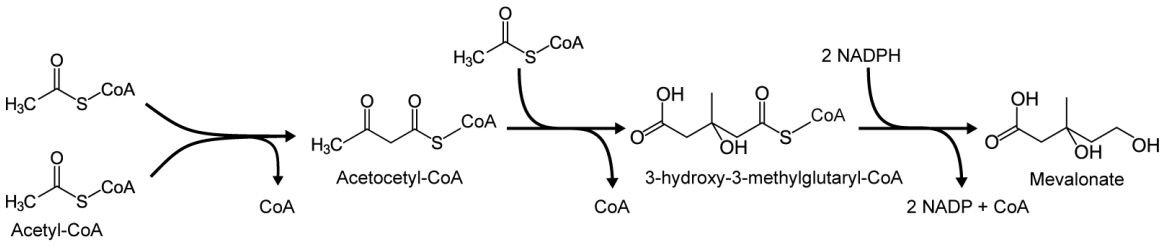


Figure 4.1: Mevalonate production pathway.

2015). GEMs for *S. cerevisiae* have been published and refined during the past 15 years (Förster et al., 2003, Mo et al., 2009, Österlund et al., 2013, Pereira et al., 2016, Zomorrodi and Maranas, 2010).

In this work, we developed a *S. cerevisiae* platform strain with enhanced mevalonate production. The strain was designed using *cameo*, a software tool for metabolic engineering (Cardoso et al., 2017). We found combinations of heterologous enzymes and gene knockouts that can increase the flux through the mevalonate pathway. We accomplish that by changing the redox balance in the cytosol. The change in redox balance was reported using a NADPH/NADP$^+$ biosensor (Zhang et al., 2016). We used the β-Carotene pathway to as an indirect measurement of increased flux through the mevalonate pathway.

Results and discussion

DifferentialFVA analysis suggests NADPH turnover as a bottleneck of mevalonate production

The DifferentialFVA method in *cameo* identifies significant flux changes required to increase flux towards desired products. The results suggested redirecting fluxes to produce more NADPH. The conversion of glycerol into dihydroxyacetone (DHA) generates extra NADPH when converted. At the cost of one ATP, the dihydroxyacetone kinase (*YML070W* or *YFL053W*) converts the DHA into DHA-phosphate. The DHA-phosphate can be incorporated into glycolysis again. Indeed, a recently published study shows that *Escherichia coli* strains producing isopentanol, limonene and bisabolene using the mevalonate pathway have a higher NADPH turnover

(Brunk et al., 2016).

Strains containing heterologous genes have the potential to produce more mevalonate

The maximum theoretical yield of mevalonate increases after inserting the new genes in the model (Figure 4.2). The production envelope of the mutant shows that at any growth rate, the amount of mevalonate produced by the mutant strain is higher than the wild-type (Figure 2). This means that the availability of NADPH is limiting the production of mevalonate.

Genetically modified strains produce more β-Carotene

We implement our designs into a *S. cerevisiae* strains containing the β-Carotene pathway. We observed improved production of β-Carotene, which has an orange color (4.3A). β-Carotene is produced from farnesyl diphosphate (FPP) which is produced from mevalonate. Cells producing more β-Carotene display brighter orange color. Because β-Carotene is produced via the mevalonate pathway, it can be used as a proxy for the mevalonate production.

In addition, the ratio between NADPH/$NADP^+$ was measured, based on the previously published biosensor (see Material and Methods). We cloned the senor in a wild-type and two mutant strains with a clean background (i.e., without the β-Carotene pathway). We saw a decrease in the NADPH/$NADP^+$ ratio (Figure 4.3B). This suggests that our mutant strains are converting more $NADP^+$ into NADPH, but the mevalonate pathway is not able to convert an equivalent amount of NADPH into $NADP^+$.

Materials and Methods

Yeast metabolic model

We performed all simulations using the *S. cerevisiae* GEM iMM904 (Mo et al., 2009). The model was download from the BiGG Models database (King et al., 2016) and follows the Constraint-Based Reconstruction and Analysis (COBRA) standards (Thiele and Palsson, 2010), therefore it is fully compatible with the software we used (see below).

In silico design of the strains

We used cameo software package (Cardoso et al., 2017) to design the strains *in silico*. Maximum theoretical yields at different growth rates was computed using Flux Balance Analysis (FBA). The identification of the required modifications was developed in 2 steps. First, we used DifferentialFVA to identify which fluxes needed to increase and decrease in relation to the wild type and identified the cofactor limitation. Second, we used the model to test different modifications in the central carbon metabolism and the effects of adding heterologous genes.

Strains and media

We the *S. cerevisiae* CEN.PK strains in these work. Cells were grown in Petri dishes on standard synthetic complete (SC) and yeast extract peptone dextrose (YPD) (2% glucose) media and incubated at 30 °C.

β-Carotene pathway

We used a strain containing the β-Carotene pathway from previously published work (Jakočiunas et al., 2015).

NADP-biosensor

We used a NADPH/NADP$^+$ redox biosensor engineered from the *Yap1p* transcription factor. The sensor is highly specific to NADPH but not to NADH Zhang et al. (2016). When the amount of NADPH over NADP increases, the strains are less fluorescence. We measure fluorescence as described in the original paper.

Conclusions

The initial attempt to engineer the central carbon metabolism and redirect flux towards the desired product has shown promising results. We observed higher β-carotene levels in the engineered strains. We also observed an increasing amount of NADPH levels in the engineered strains.

Still, we need to perform further optimization of these strains. We can use combinatorial assembly to optimize the insertion of heterologous genes and find the optimal expression levels. Plus, the mevalonate is self-regulated [i.e., subject of negative feedback by the pathway products (Dimster-Denk et al., 1994)]. We need to optimize enzymes in the mevalonate pathway too. Adding pathways upstream of mevalonate, such as the β-Carotene pathway, allows us to divert flux from the native metabolic pathways and provides a visual reporting system.

Finally, mevalonate with is a biomass precursor. Redirecting flux from that pathway competes with cellular growth and makes growth-coupled designs more difficult to predict *in silico*. We will combine computational methods to identify key genetic interventions that can reduce the growth rate and redirect more carbon towards mevalonate.

Acknowledgments

We thank Emre Özdemir for valuable discussion about constraint-based modeling in yeast. We also thank Jie Zhang for providing the plasmids with the NADPH/NADP$^+$ sensor. This work was funded by the Novo Nordisk Foundation.

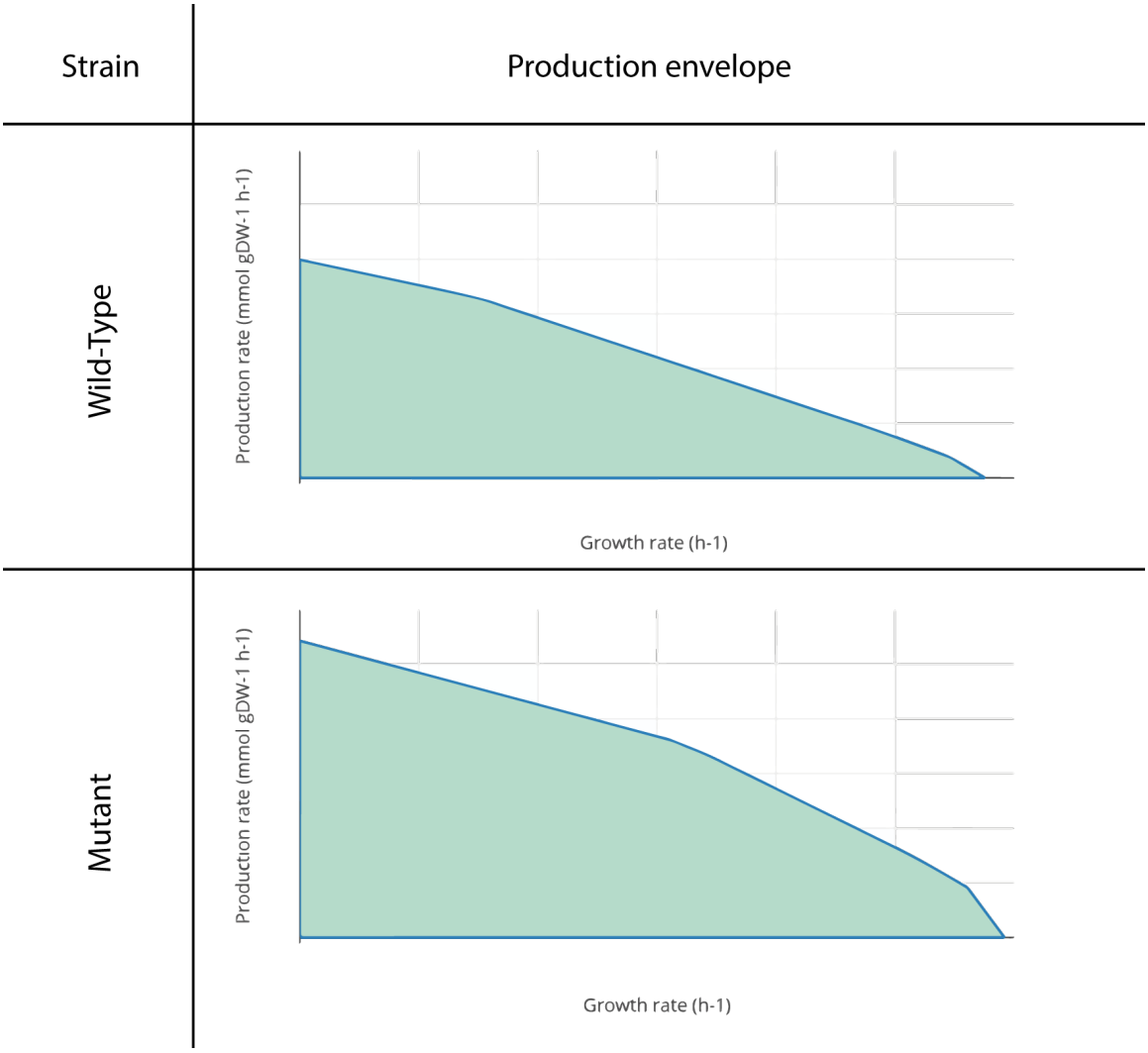


Figure 4.2: Production envelopes for mevalonate. The mutant strain has an increased the mevalonate production capacity.

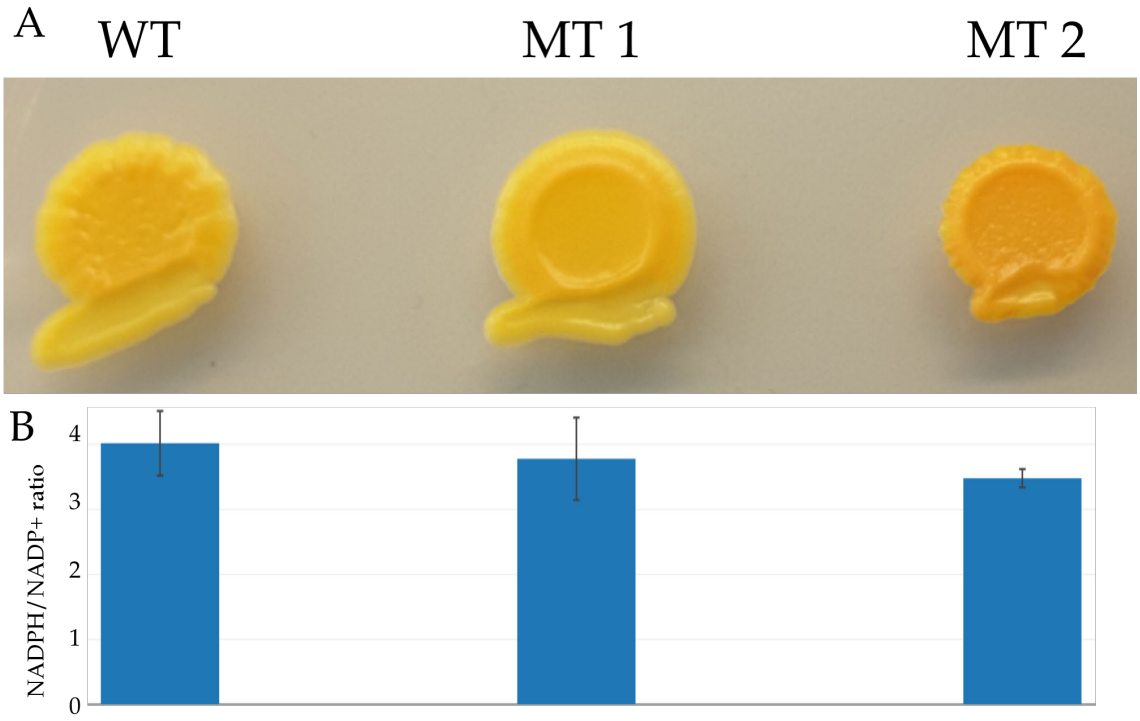


Figure 4.3: β-Carotene production in different strains and NADPH/NADP$^+$ biosensor results A) The wild-type (WT) strain with the β-Carotene pathway displays a yellow color. The mutant 2 (MT2) strain displays a orange color, which indicates higher amounts of β-Carotene per cell. B) The ratio between NADPH/NADP$^+$ decreases when more genetic modifications are applied.

References

Elizabeth Brunk, Kevin W. George, Jorge Alonso-Gutierrez, Mitchell Thompson, Edward Baidoo, George Wang, Christopher J. Petzold, Douglas McCloskey, Jonathan Monk, Laurence Yang, Edward J. O'Brien, Tanveer S. Batth, Hector Garcia Martin, Adam Feist, Paul D. Adams, Jay D. Keasling, Bernhard O. Palsson, Taek Soon Lee, Edward J. O'Brien, Tanveer S. Batth, Hector Garcia Martin, Adam Feist, Paul D. Adams, Jay D. Keasling, Bernhard O. Palsson, and Taek Soon Lee. Characterizing Strain Variation in Engineered E. coli Using a Multi-Omics-Based Workflow. *Cell Systems*, 2(5):335–346, may 2016. ISSN 24054712. doi: 10.1016/j.cels.2016.04.004.

João G R Cardoso, Kristian Jensen, Christian Lieven, Anne Sofie Lærke Hansen, Svetlana Galkina, Moritz Beber, Emre Ozdemir, Markus J. Herrgård, and Nikolaus Sonnenschein. Cameo : A Python Library for Computer Aided Metabolic Engineering and Optimization of Cell Factories. *bioRxiv*, 9:2013–2018, 2017. doi: 10.1101/147199.

Dago Dimster-Denk, Mary K Thorsness, and Jasper Rine. Feedback regulation of 3-hydroxy-3-methylglutaryl coenzyme A reductase in Saccharomyces cerevisiae. *Molecular biology of the cell*, 5(6):655–65, 1994. ISSN 1059-1524.

Jochen Förster, Iman Famili, Patrick Fu, Bernhard Ø Palsson, and Jens Nielsen. Genome-scale reconstruction of the Saccharomyces cerevisiae metabolic network. *Genome research*, 13(2): 244–253, feb 2003. ISSN 1088-9051. doi: 10.1101/gr.234503.

Tadas Jakočiunas, Arun S. Rajkumar, Jie Zhang, Dushica Arsovska, Angelica Rodriguez, Christian Bille Jendresen, Mette L. Skjødt, Alex T. Nielsen, Irina Borodina, Michael K. Jensen, and Jay D. Keasling. CasEMBLR: Cas9-Facilitated Multiloci Genomic Integration of in Vivo Assembled DNA Parts in Saccharomyces cerevisiae. *ACS Synthetic Biology*, 4(11):1126–1134, 2015. ISSN 21615063. doi: 10.1021/acssynbio.5b00007.

Zachary A. King, Justin Lu, Andreas Dräger, Philip Miller, Stephen Federowicz, Joshua A. Lerman, Ali Ebrahim, Bernhard O. Palsson, and Nathan E. Lewis. BiGG Models: A platform for integrating, standardizing and sharing genome-scale models. *Nucleic Acids Research*, 44(D1): D515–D522, jan 2016. ISSN 0305-1048. doi: 10.1093/nar/gkv1049.

Pan Liao, Andréa Hemmerlin, Thomas J. Bach, and Mee Len Chye. The potential of the mevalonate pathway for enhanced isoprenoid production. *Biotechnology Advances*, 34(5):697–713, 2016. ISSN 07349750. doi: 10.1016/j.biotechadv.2016.03.005.

Monica L Mo, Bernhard Ø Palsson, and Markus J Herrgård. Connecting extracellular metabolomic measurements to intracellular flux states in yeast. *BMC Systems Biology*, 3:37, jan 2009. ISSN 1752-0509. doi: 10.1186/1752-0509-3-37.

Elke Nevoigt. Progress in metabolic engineering of *Saccharomyces cerevisiae*. *Microbiology and molecular biology reviews : MMBR*, 72(3):379–412, 2008. ISSN 1098-5557. doi: 10.1128/MMBR.00025-07.

Edward J. O'Brien, Jonathan M. Monk, and Bernhard O. Palsson. Using genome-scale models to predict biological capabilities. *Cell*, 161(5):971–987, 2015. ISSN 10974172. doi: 10.1016/j.cell.2015.05.019.

Tobias Österlund, Intawat Nookaew, Sergio Bordel, and Jens Nielsen. Mapping condition-dependent regulation of metabolism in yeast through genome-scale modeling. *BMC systems biology*, 7:36, 2013. ISSN 1752-0509. doi: 10.1186/1752-0509-7-36.

Rui Pereira, Jens Nielsen, and Isabel Rocha. Improving the flux distributions simulated with genome-scale metabolic models of Saccharomyces cerevisiae. *Metabolic Engineering Communications*, 3:153–163, 2016. ISSN 22140301. doi: 10.1016/j.meteno.2016.05.002.

Ines Thiele and Bernhard Ø Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nature protocols*, 5(1):93–121, jan 2010. ISSN 1750-2799. doi: 10.1038/nprot.2009.203.

Fuzhong Zhang, Sarah Rodriguez, and Jay D. Keasling. Metabolic engineering of microbial pathways for advanced biofuels production. *Current Opinion in Biotechnology*, 22(6):775–783, dec 2011. ISSN 09581669. doi: 10.1016/j.copbio.2011.04.024.

Jie Zhang, Nikolaus Sonnenschein, Thomas P.B. Pihl, Kasper R. Pedersen, Michael K. Jensen, and Jay D. Keasling. Engineering an NADPH/NADP+ Redox Biosensor in Yeast. *ACS Synthetic Biology*, 5(12):1546–1556, 2016. ISSN 21615063. doi: 10.1021/acssynbio.6b00135.

Ali R Zomorrodi and Costas D Maranas. Improving the iMM904 S. cerevisiae metabolic model using essentiality and synthetic lethality data. *BMC Systems Biology*, 4(1):178, jan 2010. ISSN 1752-0509. doi: 10.1186/1752-0509-4-178.

This preservation of favourable variations and the destruction of injurious variations, I call Natural Selection, or the Survival of the Fittest. Variations neither useful nor injurious would not be affected by natural selection and would be left a fluctuating element.

Charles Darwin

5

Analysis of genetic variation and potential applications in genome-scale metabolic modeling

Summary

The second chapter of this thesis is a review about the effect of genetic variation in cell factory optimization. The tools used to evaluate genetic variants are reviewed here. However, most of these tools have been developed for medical applications and predict disease related effects. This review also exposits the potential applications of combining re-sequencing data with genome-scale metabolic models. This chapter has been published in Frontiers in Bioengineering and Biotechnology on February 2015 (https://doi.org/10.3389/fbioe.2015.00013).

Abstract

Genetic variation is the motor of evolution and allows organisms to overcome the environmental challenges they encounter. It can be both beneficial and harmful in the process of engineering cell factories for the production of proteins and chemicals. Throughout the history of biotechnology, there have been efforts to exploit genetic variation in our favor to create strains with favorable phenotypes. Genetic variation can either be present in natural populations or it can be artificially created by mutagenesis and selection or adaptive laboratory evolution. On the other hand, unintended genetic variation during a long term production process may lead to significant economic losses and it is important to understand how to control this type of variation. With the emergence of next-generation sequencing technologies, genetic variation in microbial strains can now be determined on an unprecedented scale and resolution by re-sequencing thousands of strains systematically. In this article, we review challenges in the integration and analysis of large-scale re-sequencing data, present an extensive overview of bioinformatics methods for predicting the effects of genetic variants on protein function, and discuss approaches for interfacing existing bioinformatics approaches with genome-scale models of cellular processes in order to predict effects of sequence variation on cellular phenotypes.

Keywords

Genetic Variation, SNP, Next-Generation Sequencing, Constraint-based modeling, Metabolic Engineering, Adaptive Laboratory Evolution, Metabolism, High-throughput Analysis.

Introduction

Genetic engineering has been used for several decades to manipulate microorganisms in order to allow production of valuable products, including primary metabolites (e.g., amino-acids and organic acids), secondary metabolites (e.g., antibiotics), and enzymes or other recombinant proteins (Adrio and Demain, 2010). Genetic engineering is thus a central part in the quest to establish sustainable and efficient processes for the production of fuels, chemicals, food ingredients, and pharmaceutical products.

Most of these achievements would not been possible without sequencing technologies that allowed us to identify the genetic sequences and validate the genetic manipulations in microorganisms. More recently, Next-Generation Sequencing (NGS) technologies have provided us with the capability of fast and cheap sequencing of DNA at an unprecedented scale. NGS has allowed *de novo* assembly of the genomes of thousands of organisms for which no genome sequences were previously available, ranging from complex multicellular organisms (Kelley et al., 2014, Li et al., 2010, Nakamura et al., 2013, Pegadaraju et al., 2013) to microorganisms (Soares-Castro and Santos, 2013, Yamamoto et al., 2014). NGS technologies also provide us with the means to re-sequence organisms (Atsumi et al., 2010, Wang et al., 2014), i.e., the sequencing of genetically distinct strains that are close enough to a reference strain with a sequenced genome. Re-sequencing is used to determine genetic variants ranging from single nucleotide variation (SNV) to more complex structural variants such as large deletions, inversions and translocations. The falling cost of sequencing allows routine re-sequencing of strains isolated from the wild, monitoring the genetic stability of production strains during genetic engineering and fermentation processes, and determining the genetic basis of adaptive laboratory evolution (ALE) (Herrgård and Panagiotou, 2012). In addition to biotechnological applications, re-sequencing of microbial strains plays also a key role in other areas such as epidemiology of infectious diseases caused by bacterial and fungal pathogens, and in understanding the effects of human activity on microbial diversity and evolution in the environment.

Genome-scale metabolic models GSMs, consisting of biochemical reactions and their relations to the genome and proteome of a cell (through gene-protein-reaction associations; GPR), are a proven framework for the *in silico* analysis of the metabolic physiology of microbes. GSMs have also been used successfully for the design of metabolically engineered strains with improved

production of commercially valuable proteins and metabolites: recombinant antibodies, food additives (e.g., vanillin), organic acids, ethanol, among others (Brochado et al., 2010, Tepper and Shlomi, 2009). These models have become increasingly popular over the past decade, and more than one hundred models for different organisms have been published up to this date (`http://optflux.org/models`). The greatest strength of GSMs lie in their simplicity and computational efficiency; new GSMs can be readily built from genomic annotations complemented with limited experimental data, and predictions from GSMs can be obtained using standard mathematical optimization methods (Segrè et al., 2002, Shlomi et al., 2005, Varma and Palsson, 1993) allowing phenotypic predictions within minutes.

Genetic variation that entails a complete loss of function—commonly referred to as gene knockout—has been successfully used to tailor GSMs to a specific genotype to improve the production of valuable compounds (e.g., biobutanol (Lee et al., 2008), sesquiterpene (Asadollahi et al., 2009), vanillin (Brochado et al., 2010), polyhydroxyalkanoates (Puchałka et al., 2008) or L-valine (Park et al., 2007)), but so far no methodological framework has been developed that would allow the incorporation of other types of genetic variants systematically. In this work, we review existing tools for analyzing genetic variants that capture more subtle changes such as synonymous and non-synonymous SNVs in coding regions or variants in promoter or other regulatory regions. We will focus on outlining the challenges of combining more subtle genetic variant information with GSMs in order to use models to predict strain-specific phenotypes.

Unveiling the effects of genetic variation

Genetic variability

Genetic variants, including SNVs and larger structural variants are commonly seen when natural or engineered strains are re-sequenced (Figure 5.1). SNVs can be found across the genome in different functional regions: (i) protein coding sequences, (ii) promoters and other regulatory elements such as ribosome binding sites, (iii) splice sites and other regions affecting transcript structures, and (iv) other genomic regions with unknown direct connections to any given protein function. Moreover, insertions or deletions of nucleotides (indels) within a coding region can cause a shift in the open reading frame usually denoted as frameshift mutations (Figure 5.1A). At

the genome structure level, chromosomal rearrangements, e.g., swaps, inversions, deletions, and insertions, can affect the function of one or more proteins (Figure 5.1B).

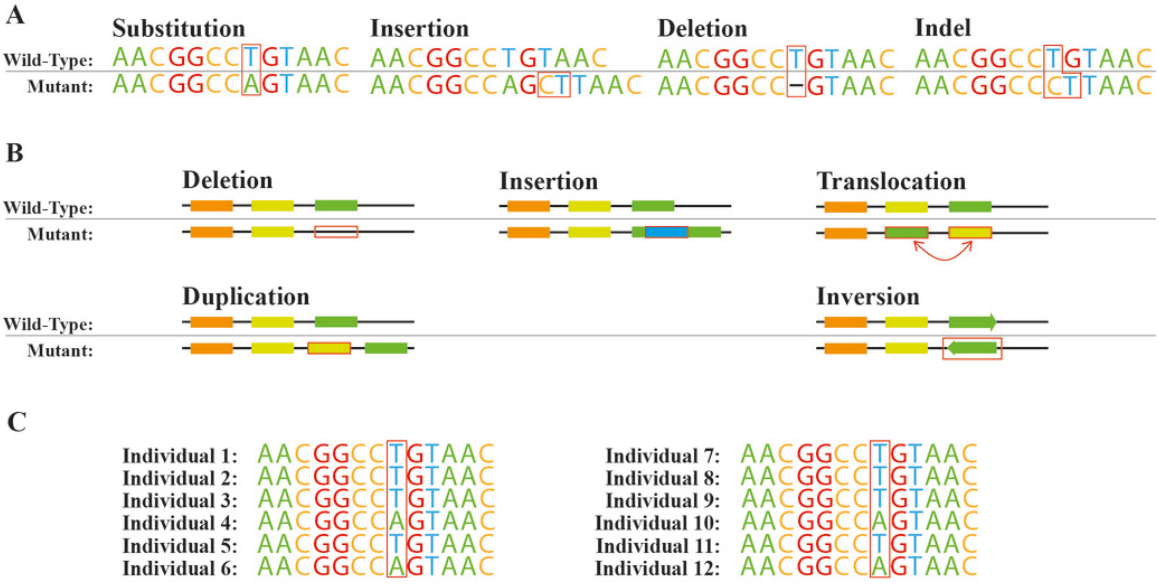


Figure 5.1: Common genetic variations. Variations at the (A) nucleotide level and (B) structural level. (C) Single nucleotide polymorphism A/T across a population.

The spectrum of the resulting effects caused by these genetic variations on individual gene or protein function or expression is very broad. Non-synonymous SNVs or in-frame indels in protein coding sequences can disrupt, enhance, or modify the activity of the protein depending on the exact amino-acid change introduced. Introduction or removal of a stop codon by specific SNVs or out-of-frame indels would be expected to result in more drastic changes of protein function. For example, the appearance of a stop codon might lead to the separation of a multi-domain protein to multiple individual single-domain proteins. The removal or replacement of a stop codon could cause translational read-through leading to an elongated protein with potential new functions (Long et al., 2003). SNVs and indels in regulatory regions such as promoters can affect the transcription or translation processes giving rise to variation in expression levels in specific proteins. In eukaryotes, variants within introns can also affect transcript structures by introducing new exons or removing existing ones. Some variations can also be completely silent with no change of phenotype, for example, a small change in stop codon location might not change the protein activity. Ideally, we should be able to predict the degree in which single

and multiple genetic variants within or near a coding locus affect the relevant protein function or expression. This would allow us to rapidly make sense of the vast quantities of re-sequencing data that is becoming available without having to test the effects of all variants experimentally.

Larger-scale structural variations, such as duplications, deletions, translocations, and inversions, can have significant effects on the expression or activity of individual proteins. For example, there can be a complete loss of one or more genes, or a duplication of genomic regions can modify the expression of multiple genes within or nearby these regions (Blount et al., 2012). Very large scale genomic changes, such as duplication of entire chromosomes, can change the activity of hundreds of proteins at once and have been reported in both natural microbial strains (Gordon et al., 2009) and in strains created by ALE (Caspeta et al., 2014). The effects of structural genomic variation are often more systemic than the effects of smaller scale variations, but any framework attempting to predict the phenotypic effects of genetic variation needs to consider both small- and large-scale variation.

In silico: predicting the effect of genetic variants

A major challenge to understanding the phenotypic consequences of genetic variation lies in our ability to predict the mechanistic consequences of mutations. Proteins are very complex structures that fall into different functional categories and can be characterized by many distinct properties. For example, how protein activities are measured depends on their functional category: transcription factors can be characterized by their binding strength to a certain promoter region while metabolic enzymes would typically be characterized by their catalytic activity and specificity for a certain substrate. Moreover, proteins don't operate in isolation but interact with each other and with metabolites, and these interactions have consequences on the activities of proteins. Here we provide a non-exhaustive review of the types of methods that are commonly used to predict the effects of genetic variants on protein function.

The study of single single nucleotide polymorphisms (SNPs) that affect human health is one of the major focus areas of modern medical research. In human genetics, SNPs are single nucleotide substitutions found in more than 1% of a population. Several algorithms were implemented to determine the effect of SNP, mostly specialized to the analysis of human genotyping data (see Table 5.1 and Figure 5.2). One limitation of most of these algorithms is that they are

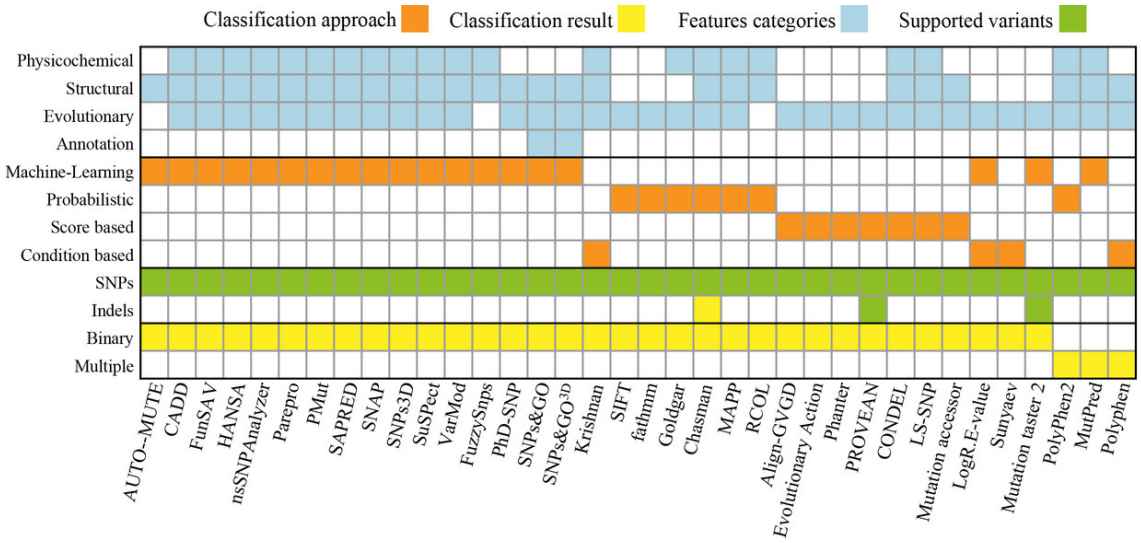


Figure 5.2: Summary of properties and approaches for variant-effect prediction software. Summary of properties and approaches for software listed in Table 5.1. The approaches found fall into 4 different categories: *Machine-Learning*, *Probabilistic*, *Score* (calculating a summarizing score of a set of hand-picked statistics), and *Rule* (using a set of empirically derived rules). These approaches provide one of two types of classifications each: a binary classification (e.g., neutral or deleterious) or a multi-classification (e.g., benign, neutral, deleterious). The features used by those approaches can be computed based on properties of the following 5 categories: (i) physicochemical properties (e.g., solvent accessibility, polarity, charge, disorder, Grantham), (ii) structural information about the primary, secondary, and tertiary structure of a protein (e.g., α-helices, β-sheets, coil), (iii) evolutionary properties (multiple sequence alignments, position-specific scoring matrices, Hidden Markov models), and (iv) genome annotation (GO terms or other protein function annotations). The supported variants were determined either by accessing the tools' websites or by the description of the approach itself.

binary classifiers—deleterious or neutral, disease causing or neutral, tolerant or intolerant. This means that the genetic changes will either be predicted to have no effect or to cause some measurable, negative impact on the phenotype. This may not be an issue in the context of human diseases as SNPs data are primarily used in diagnostics. However, fine tuning engineered microbial strains requires more than a black and white approach for predicting variant effects on protein function. This is because many genetic variants can yield proteins with either increased or decreased activity, requiring methods that are able to predict also potential gains or modifications of functions. In particular, when mutagenesis and selection or ALE methods are applied, one commonly sees gain of function mutations of specific genes that are crucial for the adaptation to for example new

carbon sources (Conrad et al., 2011).

Of the existing algorithms (Table 5.1), *SIFT* (Sorting Tolerant from Intolerant) (Ng and Henikoff, 2001) is often used as a gold standard to compare the performance of new algorithms or as a foundation for novel prediction strategies. SIFT and related approaches are based on the notion that evolutionary conservation can be used to predict the functional importance of a each amino-acid in a protein and the impact of specific amino-acid substitutions. These methods typically use multiple sequence alignments of related proteins to determine a probabilistic description of what amino-acid substitutions are allowed in specific sites within the target protein. These descriptions can be used to determine the probability that non-synonymous coding SNPs observed in a re-sequencing data set will be tolerated by the protein; substitutions with a probability score smaller than a threshold are assumed to be deleterious (Kumar et al., 2009).

SIFT provides only a binary deleterious/non-deleterious classification, and other methods have been developed to allow predicting cases where SNPs improve protein function. The *Polyphen* (Ramensky, 2002) and *PolyPhen2* (Adzhubei et al., 2010) approaches provide the means to discriminate three states when analyzing the effect of a SNP: benign, neutral, or deleterious. *Polyphen* uses a list of predetermined rules that combine the output of multiple algorithms using combinations of structural and sequence-based measures of mutation impact. *PolyPhen2* uses a machine-learning approach (a naive Bayes model) to predict an overall score for the variant effect, and the classification to three categories is based on thresholds. Although the algorithm is trained with human datasets, similar methods could potentially be used to build predictive models for variant effects in microorganisms. The overall variant effect score could also be exploited in more advanced methods that combine scores from different variants affecting different proteins to make phenotypic predictions.

Most studies on genetic variation focus on SNPs and disregard indels, which are also commonly observed when related microbial strains are compared to each other. The *PROVEAN* (Choi et al., 2012) and *Mutation taster 2* (Schwarz et al., 2014) approaches are capable of analyzing both SNPs and indels. *PROVEAN* uses substitution matrix scores (i.e. BLOSUM62) with gap and extension penalties to compute a variation score between the wild-type and mutant. More recently, *Mutation taster 2* computes several features (structural and evolutionary properties) for the mutated sequence using a Bayes classifier.

One possible approach for improving our ability to predict variant effects on protein function

would be to predict effects of amino-acid changes on protein stability and folding (Khan and Vihinen, 2010). There are a number of tools available for these tasks (Khan and Vihinen, 2010), and stability predictions could be used to predict variant effects on protein function, as strongly destabilizing mutations would result in complete loss of function for the protein. Methods for predicting variant effects on protein stability have only been found to be moderately accurate in independent evaluation studies (Khan and Vihinen, 2010). For this reason, stability predictors should be combined with other variant effect prediction approaches to improve their predictive power for general variant effect analysis. The application of these types of stability prediction methods will be discussed in section 5 in more detail together with the applications of metabolic modeling.

The majority of algorithms (53%) for variant effect prediction listed in Table 5.1 rely on machine-learning approaches (e.g., AUTO-MUTE (Masso and Vaisman, 2010), FunSAV (Wang et al., 2012) or HANSA (Acharya and Nagarajaram, 2011)), which is a practical strategy given the huge amount of data available for human diseases. Regarding the selection of features, most methods use evolutionary conservation information (92%) and more than half rely on structural properties (69%). The selection of sufficient features is a challenge in itself; no matter what approach is used, it is necessary to define which properties and attributes of proteins are capable of discriminating the phenotypes of interest. The improvements in the prediction capabilities provided by sequence-, evolution-, or structural-based features has been previously studied, and these studies have shown that the inclusion of structural properties leads to significant improvements in predictive power (Saunders and Baker, 2002). This has been recently confirmed by a benchmark performance test that includes several of the existing algorithms (Thusberg et al., 2011). Another effort to benchmark and improve different approaches is the Critical Assessment of Genome Interpretation (CAGI) community, that organizes a benchmark competition on predicting the effect of genetic variants on known disease phenotypes.

While the majority of algorithms aim to predict variant effects on individual proteins, a different objective is followed by the SNP-IN method that predicts how protein-protein interaction (PPI) are affected by a SNP (Zhao et al., 2014). This is achieved by a set of features that includes the relative free energy change between wild-type and mutant PPI, the energy of all interactions in a protein complex, and other physicochemical properties, e.g., hydrophobic solvation or water bridges. Using these features, supervised and semi-supervised machine learning

approaches are used to predict how deleterious SNPs are. This approach is a very interesting, as changes in PPIs could be used to explain epistatic interactions between multiple variants. Like some previously mentioned prediction algorithms, SNP-PI requires an existing 3D model of the protein structure and, in addition, knowledge of the PPIs a given protein is involved in.

At a larger scale, genome-wide association studies are used to identify how differences between hundreds of thousands of individuals and make genotype to phenotype consequences. This approaches work as black boxes and make use of statistical and machine-learning approaches that require huge data-sets. The current work and applications (e.g., clinical risk assessment) have been recently reviewed (Okser et al., 2014).

In vivo: deep mutational scanning and Tn-seq

Next generation sequencing has enabled studying the effects of genetic variation on individual proteins or regulatory elements *in vivo* and *in vitro*. deep mutational scanning (DMS) is an effective high-throughput method to measure the effects of mutations on protein stability and function (Fowler and Fields, 2014). The space of all possible amino-acid substitutions in a protein is exhaustively screened by first constructing a library of sequence variants using standard techniques like error prone PCR, then by using a high-throughput assay to select variants based on a fitness measure (e.g., growth rate, ligand binding or product fluorescence), and finally by applying deep sequencing to the selected and unselected sequence variant pools. This approach results in a matrix that contains fitness values for each amino-acid substitution discovered in the selected pool. Depending on the method used for creating sequence diversity and sequencing depth, DMS can also be used to measure epistatic effects between substitutions at different sites.

The applicability of DMS is primarily limited by the lack of high-throughput functional assays for most proteins, and, for example, DMS has not been applied to metabolic enzymes so far. When DMS can be applied at a broader scale, the results obtained from the assay could increase the predictive power of bioinformatic tools for genetic variation analysis by providing more complete training datasets for the types of predictive methods discussed in the previous section. Methods similar to DMS can also be used to systematically study effects of genetic variation in regulatory regions on protein expression using fluorescence protein-based assays.

Here, we will highlight a few case studies using DMS and related methods to study protein

or regulatory element function. In the analysis of *Saccharomyces cerevisiae* poly(A)-binding protein (Melamed et al., 2013), strong epistatic effects between substitutions at specific sites were discovered. Although epistasis was not widespread, this is worrying from a computational modeling perspective, as modeling approaches usually don't account for epistasis. Another important highlight is the identification of alternative start codons. Although analyzed in previous studies, the DMS has shown that some amino-acids can be replaced by methionine and yield functional proteins (Kim et al., 2013). This biological information can be extrapolated to other studies and is highly relevant when developing strategies to understand the effect of mutations, either *in vivo* or *in silico*. Strategies similar to DMS have also been used to systematically study the effects of variation in transcription factor binding sites and other regulatory elements such as ribosomal binding sites (Kosuri et al., 2013). These studies will build the foundation for predicting effects of non-coding sequence variants on protein expression.

The methods described above allow us to systematically study the effects of a large number of variants in individiual proteins or regulatory regions. In microorganisms, it is also possible to use a next generation sequencing-based method called Tn-seq to systematically study the effect of disruption of a large number of genomic loci on cellular phenotypes (van Opijnen and Camilli, 2013). Transposons are mobile DNA elements that can disrupt a genetic locus by integrating themselves into it (Figure 5.1B). Tn-seq, using high density transposon insertion libraries, can be used to interrogate the function of, for example, regulatory elements and specific protein domains in a single genome-wide assay (van Opijnen and Camilli, 2013). Tn-seq has found many applications in microbiology, and it has been used for the identification of gene function, understanding genome organization, mapping genetic interactions, or assessing gene essentiality (van Opijnen and Camilli, 2013, Yang et al., 2014). Tn-seq does not offer a resolution on the single base pair level, but the method can be rapidly used to generate sub-gene-level information relating, for example, to the essentiality of specific domains in a protein. This information in turn could be used to improve variant effect predictions, as variants in essential domains of a protein would be more likely to be predicted to be deleterious than variants in non-essential domains of the same protein.

Predicting phenotypes from genotypes at the genome-scale

Statistical and network-oriented approaches for predicting phenotypes from genotypes

Section 5 focused on the task of predicting the effects of genetic variation on individual protein function or expression. However, this is only a small part of a much larger problem, that of predicting cellular or organism phenotypic effects of all the genetic variants present in a genome. This requires combing the effects of variation on the function and expression of all proteins. So far, there have been surprisingly few efforts to take all genetic variants discovered in an individual (either a human or a microbial strain) and attempt to predict how certain phenotypes would be affected by all these variants together (Burga and Lehner, 2013, Lehner, 2013).

One of the first systematic attempts towards this goal was the pioneering study by Jelier et al. in *S. cerevisiae*, where growth phenotypes of selected yeast strains under different conditions were predicted from genetic differences between a reference strain and the strain of interest (Jelier et al., 2011). This was achieved by first predicting effects of coding and regulatory variants on protein function and expression using approaches similar to the one outlined in the previous section. These variant effect predictions were then combined into a single phenotypic prediction for the strain, using published single gene deletion growth phenotyping data for a yeast reference strain under the same condition. This approach can be considered to be highly simplistic, as the effects of multiple genetic variants acting on separate proteins were treated cumulative. Despite this, the approach still allowed accurate prediction of growth phenotypes across a broad range of conditions. There have also been a number of other approaches for predicting broader phenotypic consequences of single variants by mapping the variant data onto biological networks such as protein-protein interaction or genetic networks (Carter et al., 2013). However, these approaches have typically not attempted to use the whole genotype of an individual (i.e. more than one variant at a time) to predict specific phenotypes.

Using genome-scale metabolic models for interpreting genetic variants

The phenotype prediction methods described above are data-driven and use statistical models to predict the effects of genetic variants in the context of biological networks. However, for metabolic networks we can go beyond statistical models and graph-based descriptions to constraint-based models that are scalable to the genome-level and incorporate physicochemical, flux capacity, and reaction directionality constraints (see Price et al. (2004) for a review of constraint-based modeling). This type of mechanistic modeling approach is very useful for understanding genetic changes that affect specific metabolic phenotypes. For example, the study of SNPs that affect mitochondrial metabolism (Jamshidi and Palsson, 2006) is a good example of how variant data can be mapped onto metabolic networks in order to explain the mechanistic basis of disease phenotypes.

A genome-scale metabolic models is composed of biochemical reactions, collected from literature and the genome annotation of an organism. This system of reactions is encoded as a matrix of stoichiometric coefficients that is usually referred to as stoichiometry matrix[1]. Assuming metabolism is in a steady-state, i.e., metabolite concentrations don't change over time, all fluxes have to balance each other. These flux-balances constitute linear constraints that can easily be analyzed using methods from linear algebra.

Furthermore, after inclusion of further constraints, e.g., known uptake and secretion rates and knowledge about reaction directionality, linear optimization methods can compute biologically relevant flux vectors that maximize defined objective functions. For example, growth can be simulated by maximizing the consumption of biomass precursors in empirically determined proportions. This type of analysis is usually referred to as flux balance analysis (FBA; see Orth et al. (2010) for a comprehensive introduction to this method).

Global optimal solutions to this linear optimization problems can be calculated very efficiently using linear programming (computation times are on a millisecond to second range for genome-scale models). Thus, one can compute thousands of phenotypes in a few minutes, simply by changing the constraints of the problem (see Lewis et al. (2012) for a comprehensive list of

[1] The rows and columns of the stoichiometry matrix correspond to metabolites and reactions respectively; negative (positive) factors represent consumption (production) of substrates (products).

available *in silico* methods and Bordbar et al. (2014) for a review of their applications).

Since the relationship between reactions, enzymes, and genes (usually referred to as gene-protein-reaction (GPR) associations) is usually known and encoded in these models, the effect of a gene knockout can readily be mapped to the associated reactions by constraining their fluxes to be zero or by removal from the model. This way FBA can be used to compute the metabolic phenotype associated with a metabolic gene deletion, making it suitable for the analysis of genetic variation data that involves deletions or other mutations that lead to the complete loss of function of enzymes.

FBA assumes that knockout strains can recover to an optimal growth phenotype, which might be unrealistic in cases where regulatory mechanisms—not modeled explicitly in these models—might not be able to accommodate the desired state. Other methodologies (e.g., ROOM (Shlomi et al., 2005), MoMA (Segrè et al., 2002), MiMBl (Brochado et al., 2012) and RELATCH (Kim and Reed, 2012)) employ more plausible assumptions and have been shown to improve the accuracy of knockout predictions. For example, MoMA minimizes the euclidean distance of the wild type and mutant flux distributions, assuming that the a mutant reaches the closest feasible flux distribution that is not necessarily optimal. The predictive power of FBA and these other approaches have been extensively assessed using genome-wide gene knockout assays (Snitkin et al., 2008) and transposon insertion libraries (Yang et al., 2014) and have resulted generally in a high degree of accuracy (Monk and Palsson, 2014).

Constraint-based models have also been applied to predict epistatic interactions by simulating effects of pairwise gene deletions, but with a significantly reduced accuracy in comparison to single deletions (Szappanos et al., 2011). Furthermore, simulations of multiple gene deletions have been successfully applied in developing design strategies for metabolic engineering by redirecting flux to desired products (Blazeck and Alper, 2010, Milne et al., 2009).

A number of limiting factors can diminish the ability of constraint-based models to predict phenotypic effects of loss of function mutations: (i) missing reactions and erroneous GPRs, (ii) erroneous flux constraints due to the lack of thermodynamic or regulatory information, and (iii) the assumption of a fixed biomass composition that is known to change across growth conditions. Even with these limitations, constraint-based models still outperform statistical models in predicting consequences of gene deletions (Szappanos et al., 2011).

Since constraint-based models have demonstrated good ability to predict phenotypic out-

comes of single and multiple gene deletions, these models should also be useful for predicting effects of other genetic variants. A SNV or indel that is predicted to reduce the maximal flux rate of an enzyme can be used to constrain the upper bound of a flux. FBA and similar methods can be used to compute the effects of these variations on the phenotype, providing a system-wide overview of the effects caused by the substitution (Jamshidi et al., 2007). This is a fast and effective way of predicting phenotypes, but it requires that one can estimate the effect the variant has on the maximum flux rate. Nevertheless, cases of complete loss of function fall into the same category as gene knockouts, and combining the bioinformatic prediction tools discussed in section 5 with modeling capabilities can be used to integrate variant data. This approach can also be extended to any number of variants and genes, with the caveat that epistatic interactions are currently not captured accurately by the models.

There is currently only a limited number of studies that use GSMs to systematically explore the effects of genetic variants on phenotypes. Chang et al. (2013) conducted a study where GSMs coupled with protein structures of metabolic enzymes (GEM-PRO[2]) were used to interpret genetic variant data of *Escherichia coli* strains evolved to tolerate high temperatures (Chang et al., 2013). In this study, a GSM of *E. coli* was constrained using experimentally or bioinformatically determined thermostabilities of metabolic enzymes. Since the maximum flux capacity of a reaction is proportional to the concentration of active enzyme, temperature changes can be modeled by varying the flux constraints accordingly. This enables the prediction of enzymatic steps that are disproportionately temperature sensitive. For the evolved strains, flux balance analysis was used to explore the adaptation of the mutated enzymes; constraints associated with mutated proteins were relaxed to explain the experimentally measured growth rates (Chang et al., 2013). The study did not include separate predictions of variant effects on protein function, but rather treated all variants observed in a protein as potentially affecting its activity.

A more recent study by Nam et al. (2014) describes the use of GSMs for understanding the metabolic effects of cancer mutations. In particular, Nam *et al.* use genetic mutation information, gene expression profile data, and a human GSM (Thiele et al., 2013) to construct context-specific models for different cancer types. Loss and gain of function were systematically analyzed. Loss of function was modeled as described above (i.e. constraining affected reactions' fluxes to 0). Gain of a function, on the other hand, was modeled by adding novel promiscuous activities

[2]Genome-scale metabolic models are sometimes also referred to as GEMs.

as predicted by chemoinformatic approaches. This approach allowed the prediction of potential oncometabolites.

Kinetic modeling of genetic variants

As mentioned in the previous section, constraint-based modeling does not provide any information about the dynamic behavior of a metabolic system. A full kinetic description of a biochemical reaction network can be formulated using ordinary differential equations (Heinrich and Schuster, 1996). The major advantage of using kinetic models to study effects of genetic variation lies in their ability to account for mutations affecting catalytic or regulatory sites of an enzyme, causing either a gain or loss of catalytic activity, or binding sites of allosteric regulators.

Previous studies of red blood cell metabolism provide an overview on how SNPs can alter kinetic parameters and how kinetic models can be used to explain metabolic syndromes caused by enzyme deficiencies (Jamshidi, 2002, Jamshidi and Palsson, 2009). A disadvantage of using kinetic models is that kinetic parameters are not available for most enzymes and measuring the parameters can be challenging. For this reason, building predictive genome-scale kinetic models remains a challenge (Stanford et al., 2013a). Kinetic models are a viable tool for interpreting genetic variant data only in specific cases like, for example, the red blood cell that harbors a relatively simple metabolism.

Considerations and Future directions

Methods and tools to predict the effect of genetic variants

Many approaches have been explored in the past decade to understand and analyze the effects of genetic variation. In particular, the most active field has been the application of NGS techniques to characterize of genetic variation in the context of human disease. The amount of disease related information makes machine-learning approaches very suitable for the purpose of predicting effects of single genetic variants. Since most prediction methods have been trained and tested with human data, many of the existing methods do not perform as well or are simply not suited for the analysis of microbial genetic variants.

The other area where the study of microbial genetic variation lags behind human genetics is the systematic collection of variant and phenotyping data. Efforts to collect human genotype and phenotype data in a standardized way are currently underway with databases such as dbSNP and European Variation Archive. The UniProt database also collects variants found in the proteins sequences when this information is available. Every day thousands of new environmental or pathogenic isolates and laboratory developed microbial strains are sequenced around the world, but there is no centralized repository for this data in common use. We argue that it is of utmost importance to collect genetic variant data together with associated phenotypic data in a standard way for microbes as well.

All the existing algorithms for variant effect prediction are used to classify variants to preassigned categories (for example deleterious or non-deleterious). The approaches that predict deleterious effects can already be handled as knockouts in modeling their phenotypic effects using GSMs, but more subtle effects of mutations are missed by this approach. In order to improve our ability to predict phenotypes, there is a need to move beyond classification towards quantitative measures of variant effects on individual protein function. There are numerous features related to protein function that may be relevant for predicting variant effects: evolutionary and conservation, physicochemical (e.g., charge, polarity or free energy) and structural (e.g., secondary structures, spacial distances between amino acids or B-factors).

Existing methods for predicting variant effects have been primarily focused on generic predictors for all proteins irrespective of their function (e.g., enzymes, transcription factors, transporters, chaperons, etc.) and how do they behave in their environment (i.e. interaction with other elements: proteins, metabolites, DNA, etc.). This limits the predictive power of the methods in cases where additional information is readily available such as the relatively well studied field of microbial metabolism. For example, for metabolic enzymes, information on how kinetic parameters are affected by mutations and how these parameters vary between enzymes from different species is systematically collected in databases such as BRENDA. This type of information could be used to build improved variant effect predictors specifically for metabolic enzymes.

Modeling and high-throughput data analysis

Improvements in genome-wide variant effect prediction can also come from improving or extending genome-scale modeling approaches. Recent innovations like GEM-PRO, as discussed in Section 5, fulfill the requirement of 3D protein structures to predict the effects of genetic variation at the protein level and could be used to systematically analyze the effect of genetic variation on a genome-scale for metabolism.

Approximately 10–30% of the genes encoded in a microbial genome are represented in metabolic GSMs, limiting the utility of these models for interpreting genomic variant data. Metabolic GSMs can be extended in a number of ways to increase coverage of the overall set of genes. The transcriptional regulatory network represented as interactions between transcription factors and target genes, can help extend the coverage of predictive models and can be integrated with metabolic GSMs in a number of ways (Chandrasekaran and Price, 2010, Covert et al., 2004). These integrated models have been successfully used to make phenotypic predictions.

Another recent extension of GSMs are ME-Models[3]. These models account for the entire machinery needed for gene and protein expression, providing a higher coverage of cellular functions and a higher resolution of cellular composition (O'Brien et al., 2013). ME models have also been extended further to incorporate protein translocation from the cytoplasm to the periplasm (Liu et al., 2014). Currently, most of these extensions of GSMs have only been developed for *E. coli* and significant efforts will be required to build these extended models for other bacteria as well as eukaryotic model organisms such as *S. cerevisiae*.

The development of accurate kinetic models of metabolism, which could be useful for investigating the effects of mutations on allosteric regulation and catalytic activity, is still a tedious process. These models are usually limited to small parts of metabolism focusing on central carbon metabolism (Chassagnole et al., 2002, Machado et al., 2014, Peskov et al., 2012). There are two main reasons for these limitations: the models become huge in size and kinetic information of many enzymes is still unknown. Protocols (Stanford et al., 2013b) and methodologies (Chowdhury et al., 2014) are being developed to bring kinetic modeling to the genome-scale, but the resulting models have not yet reached sufficiently mature stage for use in variant effect prediction.

In comprehensive level, a strategy for building whole-cell models by combining multiple indi-

[3]Metabolism and Expression models

vidual models of different cellular processes including cell cycle, metabolism, transcription, and transport has been proposed (Karr et al., 2012). This strategy that also allows combining models using different representations (constraint-based, kinetic, stochastic) was used to build a functioning whole-cell model of one of the simplest prokaryotes, *Mycoplasma genitalium*. Efforts towards building more complete genome-scale models of microbes will continue as more and more information is collected and computing power increases. These models will bring us closer to the goal of genome-wide prediction of phenotypes from genotyping data.

Opportunities

Genetic engineering tools, such as MAGE (Wang et al., 2009) or CRISPR/Cas9 (Xu et al., 2014), already allow us to quickly edit genomes in a precise and accurate fashion at the single base-pair resolution level at multiple loci simultaneously. These methods will allow us to map epistatic interactions of variants within a single gene and between multiple genes more comprehensively than before. On the other hand, new *in silico* tools for predicting variant effects on phenotypes outlined above open the way to a new style of modeling at the scale of single nucleotides. These new modeling tools will greatly benefit from better training datasets that can be obtained using MAGE, CRISPR/Cas9 or other genome editing methods systematically to map epistatic interactions. The application of these novel strategies provides a way to fine tune activities of proteins in the context of complete cellular networks. For example, we envision that in the future we will have predictive models of how engineering of multiple enzymes at the single amino-acid level would affect the production of a desired metabolite.

To achieve the maximum potential of genome-scale biochemical network modeling and genetic variant analysis, a link must be created between these two fields. The necessary information to connect both worlds is already there: we know the genes, the proteins, and the reactions. The major limitations are in the current methods and data sources. On the one hand we must overcome the limitations of the tools available to predict variant effects by allowing more fine grained predictions of how a variant may affect any given protein function or expression. The usage of protein folding predictions, for example, has already been established in metabolic modeling (Chang et al., 2013), and it should be possible to use tools that predict variant effects on protein stability together with genome-scale models. On the other hand, we need to improve biochemi-

cal network modeling techniques: this is a evolving field and in the past decade there have been efforts to standardize the construction of models (Thiele and Palsson, 2010) and improving prediction methods by including high-throughput data (Machado and Herrgård, 2014).

Finally, it should be acknowledged that there will always be limitations in using solely genomic variant data as the basis for making phenotypic predictions for specific strains. We may also need to measure intermediate phenotypes such as transcript, protein, or metabolite levels for these strains in order to make predictions of how a given genotype affects a specific phenotype (Burga and Lehner, 2013). Fortunately enough comprehensive multi-omic data sets are currently being collected for wild type microbial strains, allowing refinement of modeling and bioinformatic approaches for phenotypic prediction (Ishii et al., 2007, Skelly et al., 2013). Hopefully, systematizing such data sets and a concerted action between modelers, geneticists, microbiologists, and bioinformaticians will allow us to achieve the prediction of changed and novel metabolic capabilities of a microbial strain from genomic re-sequencing data.

Disclosure/Conflict-of-Interest Statement

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Acknowledgements

JGRC, MH, and NS acknowledge support by the Novo Nordisk Foundation through the Novo Nordisk Foundation Center for Biosustainability. MRA acknowledges funding from a Biotechnology-based Synthesis and Production Research grant from the Novo Nordisk Foundation.

Table 5.1: A summary of the available software tools for predicting the effect of the genetic variants.

Tool	Description	Reference
AUTO-MUTE	Uses the '4-Body Statistical Potential' to compute a set of features—based on protein 3D structure—used to train a Random Forest model to predict *neutral* or *disease* associated SNPs.	Masso and Vaisman (2010)

Table 5.1 – continued from previous page

Tool	Description	Reference
Align-GVGD	This algorithm is based on Multiple Sequence Alignment and Grantham distance to identify missense SNPs. The authors propose a measure to calculate how much the substitution changes the Grantham distance.	Tavtigian (2005)
CADD	A machine-learning approach that uses a SVM model to predict deleterious phenotypes caused by SNPs.	Kircher et al. (2014)
Chasman *et al.* 2001	A probabilistic approach to identify which SNPs have an effect on the protein function using structural and evolutionary features that compare the variation against a dataset of mutations of lac repressor and T4 lysozyme.	Chasman and Adams (2001)
CONDEL	Consensus deleteriousness provides a score computed based on the weighted average of the normalized scores of 5 different tools: LogR.E-value, MAPP, Mutation assessor, Polyphen and STIF.	González-Pérez and López-Bigas (2011)
Evolutionary Action	Evolutionary Action is a function that links genotype with phenotype using evolutionary information, by quantifying the impact of SNPs on the fitness of a population; it correlates with disease-associated mutations.	Katsonis and Lichtarge (2014)
FATHMM	Uses hidden Markov Modelss (HMMs) to obtain position-specific information. The prediction is based on the probability change of the HMM between wild type and mutant.	Shihab et al. (2012)

Table 5.1 – continued from previous page

Tool	Description	Reference
FunSAV	A Random Forest classifier for predicting deleterious SNPs. It combines properties of the mutated protein with other tools (i.e nsSNPAnalyzer, PANTHER, PhD-SNP, PolyPhen2, SIFT and SNAP).	Wang et al. (2012)
FuzzySnps	A machine-learning approach that uses a Random Forest model trained by combining '4-Body Statistical Potential' and sequence-based features to identify tolerant and intolerant SNPs.	Barenboim et al. (2008)
Goldgar *et al.* 2004	A probabilistic approach to determine if a SNP is disease-causing, which is achieved by computing the likelihood of the protein to be similar to previously classified mutated proteins in a dataset.	Goldgar et al. (2004)
HANSA	It is a machine-learning classifier that uses a SVM model to predict whether a SNP will be neutral or disease causing.	Acharya and Nagarajaram (2011)
LogR.E-value	Uses the E-value computed by the HMMER algorithm using PFAM motifs to distinguish between deleterious and neutral SNPs.	Clifford et al. (2004)
LS-SNP	A workflow/database that uses predefined rules and machine-learning (SVN) approach to systematically characterize known SNPs.	Karchin et al. (2005)
Krishnan *et al.* 2003	Two machine-learning approaches—using SVM and Decision Trees models—are used to predict the 'effect' or 'no-effect' of a SNP.	Krishnan and Westhead (2003)
MAPP	Multivariate Analysis of Protein Polymorphism uses statistical analysis to predict the deleterious effect of SNPs.	Stone (2005)

Table 5.1 – continued from previous page

Tool	Description	Reference
Mutation assessor	Predicts the degree of impact in a protein by scoring the mutation based on the impact it causes regarding the properties of a Multiple Sequence Alignment of homologous sequences.	Reva et al. (2011)
Mutation taster 2	Uses a Bayes classifier to predict disease associated effects caused by SNPs or Indels. The classifier uses a set of features that includes splicing site and polyadenylation signal information along with structural and evolutionary properties.	Schwarz et al. (2014)
MutPred	Uses a machine-learning approach to predict disease or neutral SNPs. The features used refer to a probability of loss or gain of function regarding several functional and structural properties of the encoded protein. The authors trained SVM and Random Forest models in this work.	Li et al. (2009)
nsSNPAnalyzer	Uses a Random Forest model trained with features (consisting of SIFT score and information from Multiple Sequence Alignment and protein 3D structures) to identify disease associated SNPs.	Bao et al. (2005)
Papepro	A SVM prediction model is used by the authors to separate deleterious from neutral SNPs.	Tian et al. (2007)
Panther	Using an internal database of HMM, an evolutionary score is computed and the method predicts deleterious or neutral effects with a probability attached. The cutoff can be defined by the user (default is 3).	Thomas and Kejariwal (2004)

Table 5.1 – continued from previous page

Tool	Description	Reference
PhD-SNP	This approach uses one of two SVM models: one is trained using sequence profile features and the other is trained using sequence features. The choice of which model to use is based on a preliminary decision: if the mutation exists in the homology profile, the first model is used, otherwise the prediction is done using the second model.	Capriotti et al. (2006)
PMut	Predicts pathological or neutral effects of amino-acid substitutions. The prediction model is a Neural Network using structural-, physicochemical-, and evolutionary-based features, all calculated using sequence information only (without requiring a 3D protein structure).	Ferrer-Costa et al. (2005)
Polyphen	A set of rules defined by the authors is used to predict the effect of a SNP. These rules are built based on 3 properties: PSIC score, substitution site properties, and substitution type properties. If one of the rules matches, the output can be deleterious or benign, otherwise the substitution is classified as neutral.	Ramensky (2002)
PolyPhen2	The followup version of Polyphen, uses a naive Bayes predictor to predict damaging, benign, or neutral effects of SNPs. It uses structural information if available.	Adzhubei et al. (2010)
PROVEAN	Protein Variation Effect Analyzer computes a score based on evolutionary information to predict if a genetic variant (i.e. SNP or Indel) is neutral or deleterious.	Choi et al. (2012)

Table 5.1 – continued from previous page

Tool	Description	Reference
RCOL	Applies a Bayes' formula to calculate the probability of a SNP to be deleterious. The likelihood is tested using 20 structural and physicochemical parameters.	Terp et al. (2002)
SAPRED	Using a SVM prediction model, the authors combine features computed from evolutionary, structural, and physicochemical properties to predict disease associated SNPs.	Ye et al. (2007)
SIFT	Using a PSSM, SIFT determines the probability of a substitution being tolerated in a given position.	Ng and Henikoff (2001)
SNAP	Identifies non-neutral SNPs using machine-learning approaches that combines a battery of Neural Network models.	Bromberg et al. (2008)
SNPs3D	Combines a set of features obtained from protein 3D structure and evolutionary information to predict deleterious effects using a SVM model.	Yue et al. (2006)
SNPs&GO	A machine-learning approach that includes GO annotations as features in a SVM model to predict whether a SNP is neutral or disease associated.	Calabrese et al. (2009)
SNPs&GO^{3D}	It is the successor of SNPs&GO. It includes new features obtained from protein 3D structure.	Capriotti and Altman (2011)
Sunyaev *et al.* 2001	This approach uses a set of 7 rules empirically defined by the authors to identify nsSNPs. If one of the rules is matched, then the SNP is likely to be deleterious.	Sunyaev (2001)

Table 5.1 – continued from previous page

Tool	Description	Reference
SuSPect	A SVM model implementation to predict disease phenotypes caused by SNPs. The authors started with a high number of features until they identified 9 that provided best performance.	Yates et al. (2014)
VarMode	A machine-learning approach using a SVN model to predict the effect of SNPs that includes information regarding known protein-protein interactions. It predicts non-synonymous SNPs.	Pappalardo and Wass (2014)

References

Vishal Acharya and Hampapathalu A. Nagarajaram. Hansa: An automated method for discriminating disease and neutral human nsSNPs. *Hum Mutat*, 33(2):332–337, dec 2011. doi: 10.1002/humu.21642.

Jose-Luis Adrio and Arnold L. Demain. Recombinant organisms for production of industrial products. *Bioeng Bugs*, 1(2):116–131, mar 2010. doi: 10.4161/bbug.1.2.10484.

Ivan A Adzhubei, Steffen Schmidt, Leonid Peshkin, Vasily E Ramensky, Anna Gerasimova, Peer Bork, Alexey S Kondrashov, and Shamil R Sunyaev. A method and server for predicting damaging missense mutations. *Nat Methods*, 7(4):248–249, apr 2010. doi: 10.1038/nmeth0410-248.

Mohammad A. Asadollahi, Jérôme Maury, Kiran R. Patil, Michel Schalk, Anthony Clark, and Jens Nielsen. Enhancing sesquiterpene production in *Saccharomyces cerevisiae* through in silico driven metabolic engineering. *Metab Eng*, 11(6):328–334, nov 2009. doi: 10.1016/j.ymben.2009.07.001.

Shota Atsumi, Tung-Yun Wu, Iara M P Machado, Wei-Chih Huang, Pao-Yang Chen, Matteo Pellegrini, and James C Liao. Evolution genomic analysis, and reconstruction of isobutanol tolerance in *Escherichia coli*. *Mol Syst Bio*, 6, dec 2010. doi: 10.1038/msb.2010.98.

L. Bao, M. Zhou, and Y. Cui. nsSNPAnalyzer: identifying disease-associated nonsynonymous single nucleotide polymorphisms. *Nucleic Acids Res*, 33(Web Server):W480–W482, jul 2005. doi: 10.1093/nar/gki372.

Maxim Barenboim, Majid Masso, Iosif I. Vaisman, and D. Curtis Jamison. Statistical geometry based prediction of nonsynonymous SNP functional effects using random forest and neuro-fuzzy classifiers. *Proteins*, 71(4):1930–1939, jan 2008. doi: 10.1002/prot.21838.

John Blazeck and Hal Alper. Systems metabolic engineering: Genome-scale models and beyond. *Biotechnol J*, 5(7):647–659, feb 2010. doi: 10.1002/biot.200900247.

Zachary D. Blount, Jeffrey E. Barrick, Carla J. Davidson, and Richard E. Lenski. Genomic analysis of a key innovation in an experimental *Escherichia coli* population. *Nature*, 489(7417):513–518, sep 2012. doi: 10.1038/nature11514.

Aarash Bordbar, Jonathan M Monk, Zachary A King, and Bernhard O Palsson. Constraint-based models predict metabolic and associated cellular functions. *Nat Rev Genet*, 15(2):107–120, 2014.

Ana Rita Brochado, Claudia Matos, Birger L Møller, Jørgen Hansen, Uffe H. Mortensen, and Kiran R. Patil. Improved vanillin production in baker's yeast through in silico design. *Microb Cell Fact*, 9(1):84, 2010. doi: 10.1186/1475-2859-9-84.

Ana Rita Brochado, Sergej Andrejev, Costas D. Maranas, and Kiran R. Patil. Impact of Stoichiometry Representation on Simulation of Genotype-Phenotype Relationships in Metabolic Networks. *PLoS Comput Biol*, 8(11):e1002758, nov 2012. doi: 10.1371/journal.pcbi.1002758.

Y. Bromberg, G. Yachdav, and B. Rost. SNAP predicts effect of mutations on protein function. *Bioinformatics*, 24(20):2397–2398, oct 2008. doi: 10.1093/bioinformatics/btn435.

Alejandro Burga and Ben Lehner. Predicting phenotypic variation from genotypes phenotypes and a combination of the two. *Curr Opin Biotechnol*, 24(4):803–809, aug 2013. doi: 10.1016/j.copbio.2013.03.004.

Remo Calabrese, Emidio Capriotti, Piero Fariselli, Pier Luigi Martelli, and Rita Casadio. Functional annotations improve the predictive score of human disease-related mutations in proteins. *Hum Mutat*, 30(8):1237–1244, aug 2009. doi: 10.1002/humu.21047.

E. Capriotti, R. Calabrese, and R. Casadio. Predicting the insurgence of human genetic diseases associated to single point protein mutations with support vector machines and evolutionary information. *Bioinformatics*, 22(22):2729–2734, nov 2006. doi: 10.1093/bioinformatics/btl423.

Emidio Capriotti and Russ B. Altman. Improving the prediction of disease-related variants using protein three-dimensional structure. *BMC Bioinformatics*, 12(Suppl 4):S3, 2011. doi: 10.1186/1471-2105-12-s4-s3.

Hannah Carter, Matan Hofree, and Trey Ideker. Genotype to phenotype via network analysis. *Curr Opin Genet Dev*, 23(6):611–621, dec 2013. doi: 10.1016/j.gde.2013.10.003.

L. Caspeta, Y. Chen, A. Feizi P. Ghiaci, S. Buskov, B. M. Hallstrom, D. Petranovic, and Jens Nielsen. Altered sterol composition renders yeast thermotolerant. *Science*, 346(6205):75–78, oct 2014. doi: 10.1126/science.1258137.

S. Chandrasekaran and N. D. Price. Probabilistic integrative modeling of genome-scale metabolic and regulatory networks in *Escherichia coli* and *Mycobacterium tuberculosis*. *PNAS*, 107(41): 17845–17850, sep 2010. doi: 10.1073/pnas.1005139107.

R. L. Chang, K. Andrews, D. Kim, Z. Li, A. Godzik, and Bernhard Ø. Palsson. Structural Systems Biology Evaluation of Metabolic Thermotolerance in *Escherichia coli*. *Science*, 340(6137):1220–1223, jun 2013. doi: 10.1126/science.1234012.

Daniel Chasman and R.Mark Adams. Predicting the functional consequences of non-synonymous single nucleotide polymorphisms: structure-based assessment of amino acid variation. *J Mol Biol*, 307(2):683–706, mar 2001. doi: 10.1006/jmbi.2001.4510.

Christophe Chassagnole, Naruemol Noisommit-Rizzi, Joachim W. Schmid, Klaus Mauch, and Matthias Reuss. Dynamic modeling of the central carbon metabolism of *Escherichia coli*. *Biotechnology and Bioengineering*, 79(1):53–73, jul 2002. doi: 10.1002/bit.10288.

Yongwook Choi, Gregory E. Sims, Sean Murphy, Jason R. Miller, and Agnes P. Chan. Predicting the Functional Effect of Amino Acid Substitutions and Indels. *PLoS One*, 7(10):e46688, oct 2012. doi: 10.1371/journal.pone.0046688.

Anupam Chowdhury, Ali R. Zomorrodi, and Costas D. Maranas. k-OptForce: Integrating Kinetics with Flux Balance Analysis for Strain Design. *PLoS Comput Biol*, 10(2):e1003487, feb 2014. doi: 10.1371/journal.pcbi.1003487.

R. J. Clifford, M. N. Edmonson, C. Nguyen, and K. H. Buetow. Large-scale analysis of non-synonymous coding region single nucleotide polymorphisms. *Bioinformatics*, 20(7):1006–1014, jan 2004. doi: 10.1093/bioinformatics/bth029.

Tom M. Conrad, Nathan E. Lewis, and Bernhard Ø. Palsson. Microbial laboratory evolution in the era of genome-scale science. *Mol Syst Bio*, 7(1):509–509, jan 2011. doi: 10.1038/msb.2011.42.

Markus W. Covert, Eric M. Knight, Jennifer L. Reed, Markus Herrgård, and Bernhard Ø. Palsson. Integrating high-throughput and computational data elucidates bacterial networks. *Nature*, 429(6987):92–96, may 2004. doi: 10.1038/nature02456.

C. Ferrer-Costa, J. L. Gelpi, L. Zamakola, I. Parraga, X. de la Cruz, and M. Orozco. PMUT: a web-based tool for the annotation of pathological mutations on proteins. *Bioinformatics*, 21 (14):3176–3178, jul 2005. doi: 10.1093/bioinformatics/bti486.

Douglas M. Fowler and Stanley Fields. Deep mutational scanning: a new style of protein science. *Nat Methods*, 11(8):801–807, jul 2014. doi: 10.1038/nmeth.3027.

D. Goldgar, D. Easton, A. Deffenbaugh, A. Monteiro, S. Tavtigian, and F. Couch. Integrated Evaluation of DNA Sequence Variants of Unknown Clinical Significance: Application to BRCA1 and BRCA2. *Am J Hum Genet*, 75(4):535–544, oct 2004. doi: 10.1086/424388.

Abel González-Pérez and Nuria López-Bigas. Improving the Assessment of the Outcome of Nonsynonymous SNVs with a Consensus Deleteriousness Score, Condel. *Am J Hum Genet*, 88(4):440–449, apr 2011. doi: 10.1016/j.ajhg.2011.03.004.

Jonathan L. Gordon, Kevin P. Byrne, and Kenneth H. Wolfe. Additions, losses, and rearrangements on the evolutionary route from a reconstructed ancestor to the modern *Saccharomyces cerevisiae* genome. *PLoS Genet*, 5(5):e1000485, 05 2009. doi: 10.1371/journal.pgen.1000485.

Reinhart Heinrich and Stefan Schuster. *The Regulation Of Cellular Systems*. Springer, August 1996. ISBN 0412032619.

Markus Herrgård and Gianni Panagiotou. Analyzing the genomic variation of microbial cell factories in the era of "New Biotechnology". *Comput Struct Biotechnol J*, 3(4):1–8, oct 2012. doi: 10.5936/csbj.201210012.

N. Ishii, K. Nakahigashi, T. Baba, M. Robert, T. Soga, A. Kanai, T. Hirasawa, M. Naba, K. Hirai, A. Hoque, Y. Kakazu P. Y. Ho, K. Sugawara, S. Igarashi, S. Harada, T. Masuda, N. Sugiyama, T. Togashi, M. Hasegawa, Y. Takai, K. Yugi, K. Arakawa, N. Iwata, Y. Toya, Y. Nakayama, T. Nishioka, K. Shimizu, H. Mori, and M. Tomita. Multiple High-Throughput Analyses Monitor the Response of E. coli to Perturbations. *Science*, 316(5824):593–597, apr 2007. doi: 10.1126/science.1132067.

N. Jamshidi. In Silico Model-Driven Assessment of the Effects of Single Nucleotide Polymorphisms (SNPs) on Human Red Blood Cell Metabolism. *Genome Res*, 12(11):1687–1692, nov 2002. doi: 10.1101/gr.329302.

Neema Jamshidi and Bernhard Ø Palsson. Systems biology of SNPs. *Mol Syst Bio*, 2, jul 2006. doi: 10.1038/msb4100077.

Neema Jamshidi and Bernhard Ø. Palsson. Using in silico models to simulate dual perturbation experiments: procedure development and interpretation of outcomes. *BMC Syst Biol*, 3(1): 44, 2009. doi: 10.1186/1752-0509-3-44.

Neema Jamshidi, Thuy D. Vo, and Bernhard Ø. Palsson. In Silico Analysis of SNPs and Other High-Throughput Data. In Jun Zhang and Gregg Rokosh, editors, *Cardiac Gene Expression*, volume 366 of *Methods in Molecular Biology*, pages 267–285. Humana Press, 2007. ISBN 978-1-58829-352-7. doi: 10.1007/978-1-59745-030-0_15.

Rob Jelier, Jennifer I. Semple, Rosa Garcia-Verdugo, and Ben Lehner. Predicting phenotypic variation in yeast from individual genome sequences. *Nat Genet*, 43(12):1270–1274, nov 2011. doi: 10.1038/ng.1007.

R. Karchin, M. Diekhans, L. Kelly, D. J. Thomas, U. Pieper, N. Eswar, D. Haussler, and A. Sali. LS-SNP: large-scale annotation of coding non-synonymous SNPs based on multiple information sources. *Bioinformatics*, 21(12):2814–2820, jun 2005. doi: 10.1093/bioinformatics/bti442.

Jonathan R. Karr, Jayodita C. Sanghvi, Derek N. Macklin, Miriam V. Gutschow, Jared M. Jacobs, Benjamin Bolival, Nacyra Assad-Garcia, John I. Glass, and Markus W. Covert. A Whole-Cell Computational Model Predicts Phenotype from Genotype. *Cell*, 150(2):389–401, jul 2012. doi: 10.1016/j.cell.2012.05.044.

Panagiotis Katsonis and Olivier Lichtarge. A formal perturbation equation between genotype and phenotype determines the Evolutionary Action of protein-coding variations on fitness. *Genome Res*, sep 2014. doi: 10.1101/gr.176214.114.

Joanna L. Kelley, Justin T. Peyton, Anna-Sophie Fiston-Lavier, Nicholas M. Teets, Muh-Ching Yee, J. Spencer Johnston, Carlos D. Bustamante, Richard E. Lee, and David L. Denlinger. Compact genome of the Antarctic midge is likely an adaptation to an extreme environment. *Nat Commun*, 5, aug 2014. doi: 10.1038/ncomms5611.

Sofia Khan and Mauno Vihinen. Performance of protein stability predictors. *Hum Mutat*, 31 (6):675–684, mar 2010. doi: 10.1002/humu.21242.

I. Kim, C. R. Miller, D. L. Young, and S. Fields. High-throughput Analysis of in vivo Protein Stability. *Mol Cell Proteomics*, 12(11):3370–3378, nov 2013. doi: 10.1074/mcp.0113.031708.

Joonhoon Kim and Jennifer L. Reed. RELATCH: relative optimality in metabolic networks explains robust metabolic and regulatory responses to perturbations. *Genome Biol*, 13(9):R78, 2012. doi: 10.1186/gb-2012-13-9-r78.

Martin Kircher, Daniela M. Witten, Preti Jain, Brian J O'Roak, Gregory M. Cooper, and Jay Shendure. A general framework for estimating the relative pathogenicity of human genetic variants. *Nat Genet*, 46(3):310–315, feb 2014. doi: 10.1038/ng.2892.

S. Kosuri, D. B. Goodman, G. Cambray, V. K. Mutalik, Y. Gao, A. P. Arkin, D. Endy, and G. M. Church. Composability of regulatory sequences controlling transcription and translation in *Escherichia coli*. *PNAS*, 110(34):14024–9, aug 2013. doi: 10.1073/pnas.1301301110.

V. G. Krishnan and D. R. Westhead. A comparative study of machine-learning methods to predict the effects of single nucleotide polymorphisms on protein function. *Bioinformatics*, 19(17): 2199–2209, nov 2003. doi: 10.1093/bioinformatics/btg297.

Prateek Kumar, Steven Henikoff, and Pauline C. Ng. Predicting the effects of coding non-synonymous variants on protein function using the SIFT algorithm. *Nat Protoc*, 4(8):1073–1081, jun 2009. doi: 10.1038/nprot.2009.86.

Joungmin Lee, Hongseok Yun, Adam M. Feist, Bernhard Ø. Palsson, and Sang Yup Lee. Genome-scale reconstruction and in silico analysis of the *Clostridium acetobutylicum* ATCC 824 metabolic network. *Appl Microbiol Biotechnol*, 80(5):849–862, aug 2008. doi: 10.1007/s00253-008-1654-4.

Ben Lehner. Genotype to phenotype: lessons from model organisms for human genetics. *Nat Rev Genet*, 14(3):168–178, jan 2013. doi: 10.1038/nrg3404.

Nathan E Lewis, Harish Nagarajan, and Bernhard O Palsson. Constraining the metabolic genotype–phenotype relationship using a phylogeny of *in silico* methods. *Nat Rev Microbiol*, 10(4):291–305, 2012.

B. Li, V. G. Krishnan, M. E. Mort, F. Xin, K. K. Kamati, D. N. Cooper, S. D. Mooney, and P. Radivojac. Automated inference of molecular mechanisms of disease from amino acid substitutions. *Bioinformatics*, 25(21):2744–2750, nov 2009. doi: 10.1093/bioinformatics/btp528.

Ruiqiang Li, Wei Fan, Geng Tian, Hongmei Zhu, Lin He, Jing Cai, Quanfei Huang, Qingle Cai, Bo Li, Yinqi Bai, Zhihe Zhang, Yaping Zhang, Wen Wang, Jun Li, Fuwen Wei, Heng Li, Min Jian, Jianwen Li, Zhaolei Zhang, Rasmus Nielsen, Dawei Li, Wanjun Gu, Zhentao Yang, Zhaoling Xuan, Oliver A. Ryder, Frederick Chi-Ching Leung, Yan Zhou, Jianjun Cao, Xiao Sun, Yonggui Fu, Xiaodong Fang, Xiaosen Guo, Bo Wang, Rong Hou, Fujun Shen, Bo Mu, Peixiang Ni, Runmao Lin, Wubin Qian, Guodong Wang, Chang Yu, Wenhui Nie, Jinhuan Wang, Zhigang Wu, Huiqing Liang, Jiumeng Min, Qi Wu, Shifeng Cheng, Jue Ruan, Mingwei Wang, Zhongbin Shi, Ming Wen, Binghang Liu, Xiaoli Ren, Huisong Zheng, Dong Dong, Kathleen Cook, Gao Shan, Hao Zhang, Carolin Kosiol, Xueying Xie, Zuhong Lu, Hancheng Zheng, Yingrui Li, Cynthia C. Steiner, Tommy Tsan-Yuk Lam, Siyuan Lin, Qinghui Zhang, Guoqing Li, Jing Tian, Timing Gong, Hongde Liu, Dejin Zhang, Lin Fang, Chen Ye, Juanbin Zhang, Wenbo Hu, Anlong Xu, Yuanyuan Ren, Guojie Zhang, Michael W. Bruford, Qibin Li, Lijia Ma, Yiran Guo, Na An, Yujie Hu, Yang Zheng, Yongyong Shi, Zhiqiang Li, Qing Liu, Yanling Chen, Jing Zhao, Ning Qu, Shancen Zhao, Feng Tian, Xiaoling Wang, Haiyin Wang, Lizhi Xu, Xiao Liu, Tomas Vinar, Yajun Wang, Tak-Wah Lam, Siu-Ming Yiu, Shiping Liu, Hemin Zhang, Desheng Li, Yan Huang, Xia Wang, Guohua Yang, Zhi Jiang, Junyi Wang,

Nan Qin, Li Li, Jingxiang Li, Lars Bolund, Karsten Kristiansen, Gane Ka-Shu Wong, Maynard Olson, Xiuqing Zhang, Songgang Li, Huanming Yang, Jian Wang, and Jun Wang. The sequence and *de novo* assembly of the giant panda genome. *Nature*, 463(7279):311–317, jan 2010. doi: 10.1038/nature08696.

Joanne K. Liu, Edward J. O'Brien, Joshua A. Lerman, Karsten Zengler, Bernhard Ø. Palsson, and Adam M. Feist. Reconstruction and modeling protein translocation and compartmentalization in *Escherichia coli* at the genome-scale. *BMC Syst Biol*, 8(1):110, 2014. doi: 10.1186/s12918-014-0110-6.

Manyuan Long, Esther Betrán, Kevin Thornton, and Wen Wang. The origin of new genes: glimpses from the young and old. *Nat Rev Genet*, 4(11):865–875, nov 2003. doi: 10.1038/nrg1204.

Daniel Machado and Markus Herrgård. Systematic Evaluation of Methods for Integration of Transcriptomic Data into Constraint-Based Models of Metabolism. *PLoS Comput Biol*, 10(4): e1003580, 04 2014. doi: 10.1371/journal.pcbi.1003580.

Daniel Machado, Lígia R. Rodrigues, and Isabel Rocha. A kinetic model for curcumin production in *Escherichia coli*. *Biosystems*, 125:16–21, nov 2014. doi: 10.1016/j.biosystems.2014.09.001.

Majid Masso and Iosif I. Vaisman. Knowledge-based computational mutagenesis for predicting the disease potential of human non-synonymous single nucleotide polymorphisms. *J Theor Biol*, 266(4):560–568, oct 2010. doi: 10.1016/j.jtbi.2010.07.026.

D. Melamed, D. L. Young, C. E. Gamble, C. R. Miller, and S. Fields. Deep mutational scanning of an RRM domain of the *Saccharomyces cerevisiae* poly(A)-binding protein. *RNA*, 19(11): 1537–1551, sep 2013. doi: 10.1261/rna.040709.113.

Caroline B. Milne, Pan-Jun Kim, James A. Eddy, and Nathan D. Price. Accomplishments in genome-scale in silico modeling for industrial and medical biotechnology. *Biotechnol J*, 4(12): 1653–1670, 2009. doi: 10.1002/biot.200900234.

J. Monk and Bernhard Ø. Palsson. Predicting microbial growth. *Science*, 344(6191):1448–1449, jun 2014. doi: 10.1126/science.1253388.

Y. Nakamura, K. Mori, K. Saitoh, K. Oshima, M. Mekuchi, T. Sugaya, Y. Shigenobu, N. Ojima, S. Muta, A. Fujiwara, M. Yasuike, I. Oohara, H. Hirakawa, V. S. Chowdhury, T. Kobayashi, K. Nakajima, M. Sano, T. Wada, K. Tashiro, K. Ikeo, M. Hattori, S. Kuhara, T. Gojobori, and K. Inouye. Evolutionary changes of multiple visual pigment genes in the complete genome of Pacific bluefin tuna. *PNAS*, 110(27):11061–11066, jun 2013. doi: 10.1073/pnas.1302051110.

Hojung Nam, Miguel Campodonico, Aarash Bordbar, Daniel R. Hyduke, Sangwoo Kim, Daniel C. Zielinski, and Bernhard Ø. Palsson. A Systems Approach to Predict Oncometabolites via Context-Specific Genome-Scale Metabolic Networks. *PLoS Comput Biol*, 10(9): e1003837, sep 2014. doi: 10.1371/journal.pcbi.1003837.

Pauline C. Ng and Steven Henikoff. Predicting Deleterious Amino Acid Substitutions. *Genome Res*, 11(5):863–874, may 2001. doi: 10.1101/gr.176601.

Edward J. O'Brien, Joshua A. Lerman, R. L. Chang, Daniel R. Hyduke, and Bernhard Ø. Palsson. Genome-scale models of metabolism and gene expression extend and refine growth phenotype prediction. *Mol Syst Bio*, 9(1):693–693, jan 2013. doi: 10.1038/msb.2013.52.

Sebastian Okser, Tapio Pahikkala, Antti Airola, Tapio Salakoski, Samuli Ripatti, and Tero Aittokallio. Regularized machine learning in the genetic prediction of complex traits. *PLoS Genet*, 10(11):e1004754, 11 2014. doi: 10.1371/journal.pgen.1004754.

Jeffrey D. Orth, Ines Thiele, and Bernhard Ø Palsson. What is flux balance analysis? *Nature biotechnology*, 28(3):245–8, mar 2010. ISSN 1546-1696. doi: 10.1038/nbt.1614.

Morena Pappalardo and Mark N. Wass. VarMod: modelling the functional effects of non-synonymous variants. *Nucleic Acids Res*, 42(W1):W331–W336, jun 2014. doi: 10.1093/nar/gku483.

J. H. Park, K. H. Lee, T. Y. Kim, and S. Y. Lee. Metabolic engineering of *Escherichia coli* for the production of L-valine based on transcriptome analysis and in silico gene knockout simulation. *PNAS*, 104(19):7797–7802, apr 2007. doi: 10.1073/pnas.0702609104.

Venkatramana Pegadaraju, Rick Nipper, Brent Hulke, Lili Qi, and Quentin Schultz. *De novo* sequencing of sunflower genome for SNP discovery using RAD (Restriction site Associated DNA) approach. *BMC Genomics*, 14(1):556, 2013. doi: 10.1186/1471-2164-14-556.

Kirill Peskov, Ekaterina Mogilevskaya, and Oleg Demin. Kinetic modelling of central carbon metabolism in *Escherichia coli*. *FEBS J*, 279(18):3374–3385, sep 2012. doi: 10.1111/j.1742-4658.2012.08719.x.

Nathan D Price, Jennifer L Reed, and Bernhard Ø Palsson. Genome-scale models of microbial cells: evaluating the consequences of constraints. *Nat Rev Microbiol*, 2(11):886–897, 2004.

Jacek Puchałka, Matthew A. Oberhardt, Miguel Godinho, Agata Bielecka, Daniela Regenhardt, Kenneth N. Timmis, Jason A. Papin, and Vítor A. P. Martins dos Santos. Genome-Scale Reconstruction and Analysis of the *Pseudomonas putida* KT2440 Metabolic Network Facilitates Applications in Biotechnology. *PLoS Comput Biol*, 4(10):e1000210, oct 2008. doi: 10.1371/journal.pcbi.1000210.

V. Ramensky. Human non-synonymous SNPs: server and survey. *Nucleic Acids Res*, 30(17):3894–3900, sep 2002. doi: 10.1093/nar/gkf493.

B. Reva, Y. Antipin, and C. Sander. Predicting the functional impact of protein mutations: application to cancer genomics. *Nucleic Acids Res*, 39(17):e118–e118, jul 2011. doi: 10.1093/nar/gkr407.

Christopher T. Saunders and David Baker. Evaluation of Structural and Evolutionary Contributions to Deleterious Mutation Prediction. *J Mol Biol*, 322(4):891–901, sep 2002. doi: 10.1016/s0022-2836(02)00813-6.

Jana Marie Schwarz, David N. Cooper, Markus Schuelke, and Dominik Seelow. MutationTaster2: mutation prediction for the deep-sequencing age. *Nat Methods*, 11(4):361–362, mar 2014. doi: 10.1038/nmeth.2890.

Daniel Segrè, Dennis Vitkup, and George M. Church. Analysis of optimality in natural and perturbed metabolic networks. *PNAS*, 99(23):15112–15117, nov 2002. doi: 10.1073/pnas.232349399.

Hashem A. Shihab, Julian Gough, David N. Cooper, Peter D. Stenson, Gary L. A. Barker, Keith J. Edwards, Ian N. M. Day, and Tom R. Gaunt. Predicting the Functional Molecular, and Phenotypic Consequences of Amino Acid Substitutions using Hidden Markov Models. *Hum Mutat*, 34(1):57–65, nov 2012. doi: 10.1002/humu.22225.

T. Shlomi, O. Berkman, and E. Ruppin. Regulatory on/off minimization of metabolic flux changes after genetic perturbations. *PNAS*, 102(21):7695–7700, may 2005. doi: 10.1073/pnas.0406346102.

D. A. Skelly, G. E. Merrihew, M. Riffle, C. F. Connelly, E. O. Kerr, M. Johansson, D. Jaschob, B. Graczyk, N. J. Shulman, J. Wakefield, S. J. Cooper, S. Fields, W. S. Noble, E. G. D. Muller, T. N. Davis, M. J. Dunham, M. J. MacCoss, and J. M. Akey. Integrative phenomics reveals insight into the structure of phenotypic diversity in budding yeast. *Genome Res*, 23(9):1496–1504, may 2013. doi: 10.1101/gr.155762.113.

Evan S. Snitkin, Aimée M. Dudley, Daniel M. Janse, Kaisheen Wong, George M. Church, and Daniel Segrè. Model-driven analysis of experimentally determined growth phenotypes for 465 yeast gene deletion mutants under 16 different conditions. *Genome Biol*, 9(9):R140, 2008. doi: 10.1186/gb-2008-9-9-r140.

P. Soares-Castro and P. M. Santos. Towards the Description of the Genome Catalogue of *Pseudomonas sp.* Strain M1. *Genome Announc*, 1(1):e00146–12–e00146–12, jan 2013. doi: 10.1128/genomea.00146-12.

Natalie J Stanford, Timo Lubitz, Kieran Smallbone, Edda Klipp, Pedro Mendes, and Wolfram Liebermeister. Systematic construction of kinetic models from genome-scale metabolic networks. *PLoS ONE*, 8(11):e79195, 2013a.

Natalie J. Stanford, Timo Lubitz, Kieran Smallbone, Edda Klipp, Pedro Mendes, and Wolfram Liebermeister. Systematic Construction of Kinetic Models from Genome-Scale Metabolic Networks. *PLoS ONE*, 8(11):e79195, nov 2013b. doi: 10.1371/journal.pone.0079195.

E. A. Stone. Physicochemical constraint violation by missense substitutions mediates impairment of protein function and disease severity. *Genome Res*, 15(7):978–986, jun 2005. doi: 10.1101/gr.3804205.

S. Sunyaev. Prediction of deleterious human alleles. *Hum Mol Genet*, 10(6):591–597, mar 2001. doi: 10.1093/hmg/10.6.591.

Balázs Szappanos, Károly Kovács, Béla Szamecz, Frantisek Honti, Michael Costanzo, Anastasia Baryshnikova, Gabriel Gelius-Dietrich, Martin J. Lercher, Márk Jelasity, Chad L. Myers, Brenda J. Andrews, Charles Boone, Stephen G. Oliver, Csaba Pál, and Balázs Papp. An integrated approach to characterize genetic interaction networks in yeast metabolism. *Nat Genet*, 43(7):656–662, may 2011. doi: 10.1038/ng.846.

S. V. Tavtigian. Comprehensive statistical study of 452 BRCA1 missense substitutions with classification of eight recurrent substitutions as neutral. *J Med Genet*, 43(4):295–305, sep 2005. doi: 10.1136/jmg.2005.033878.

N. Tepper and T. Shlomi. Predicting metabolic engineering knockout strategies for chemical production: accounting for competing pathways. *Bioinformatics*, 26(4):536–543, dec 2009. doi: 10.1093/bioinformatics/btp704.

Bent N. Terp, David N. Cooper, Inge T. Christensen, Flemming S. Jøgensen, Peter Bross, Niels Gregersen, and Michael Krawczak. Assessing the relative importance of the biophysical properties of amino acid substitutions associated with human genetic disease. *Hum Mutat*, 20(2): 98–109, jul 2002. doi: 10.1002/humu.10095.

Ines Thiele and Bernhard Ø. Palsson. A protocol for generating a high-quality genome-scale metabolic reconstruction. *Nat Protoc*, 5(1):93–121, jan 2010. doi: 10.1038/nprot.2009.203.

Ines Thiele, Neil Swainston, Ronan MT Fleming, Andreas Hoppe, Swagatika Sahoo, Maike K Aurich, Hulda Haraldsdottir, Monica L Mo, Ottar Rolfsson, Miranda D Stobbe, et al. A community-driven global reconstruction of human metabolism. *Nat Biotechnol*, 31(5):419–425, 2013.

P. D. Thomas and A. Kejariwal. Coding single-nucleotide polymorphisms associated with complex vs. Mendelian disease: Evolutionary evidence for differences in molecular effects. *PNAS*, 101(43):15398–15403, oct 2004. doi: 10.1073/pnas.0404380101.

Janita Thusberg, Ayodeji Olatubosun, and Mauno Vihinen. Performance of mutation pathogenicity prediction methods on missense variants. *Hum Mutat*, 32(4):358–368, feb 2011. doi: 10.1002/humu.21445.

Jian Tian, Ningfeng Wu, Xuexia Guo, Jun Guo, Juhua Zhang, and Yunliu Fan. Predicting the phenotypic effects of non-synonymous single nucleotide polymorphisms based on support vector machines. *BMC Bioinformatics*, 8(1):450, 2007. doi: 10.1186/1471-2105-8-450.

Tim van Opijnen and Andrew Camilli. Transposon insertion sequencing: a new tool for systems-level analysis of microorganisms. *Nat Rev Micro*, 11(7):435–442, may 2013. doi: 10.1038/nrmicro3033.

Amit Varma and Bernhard Ø. Palsson. Metabolic Capabilities of *Escherichia coli* II. Optimal Growth Patterns. *J Theor Biol*, 165(4):503–522, dec 1993. doi: 10.1006/jtbi.1993.1203.

Harris H. Wang, Farren J. Isaacs, Peter A. Carr, Zachary Z. Sun, George Xu, Craig R. Forest, and George M. Church. Programming cells by multiplex genome engineering and accelerated evolution. *Nature*, 460(7257):894–898, jul 2009. doi: 10.1038/nature08187.

Linhai Wang, Xuelian Han, Yanxin Zhang, Donghua Li, Xin Wei, Xia Ding, and Xiurong Zhang. Deep resequencing reveals allelic variation in Sesamum indicum. *BMC Plant Biology*, 14(1):225, 2014. doi: 10.1186/s12870-014-0225-3.

M. Wang, X.M Zhao, K. Takemoto, H. Xu, Y. Li, T. Akutsu, and J. Song. FunSAV: predicting the functional effect of single amino acid variants using a two-stage random forest model. *PLoS One*, 7:e43847, 2012. doi: 10.1371/journal.pone.0043847.

T. Xu, Y. Li, J. D. Van Nostrand, Z. He, and J. Zhou. Cas9-Based Tools for Targeted Genome Editing and Transcriptional Control. *Appl Environ Microbiol*, 80(5):1544–1552, jan 2014. doi: 10.1128/aem.03786-13.

K. Yamamoto, H. Tamaki, H. Cadillo-Quiroz, H. Imachi, N. Kyrpides, T. Woyke, L. Goodwin, S. H. Zinder, Y. Kamagata, and W.-T. Liu. Complete Genome Sequence of *Methanoregula formicica* SMSPT a Mesophilic Hydrogenotrophic Methanogen Isolated from

a Methanogenic Upflow Anaerobic Sludge Blanket Reactor. *Genome Announc*, 2(5):e00870–14–e00870–14, sep 2014. doi: 10.1128/genomea.00870-14.

Hong Yang, Elias W. Krumholz, Evan D. Brutinel, Nagendra P. Palani, Michael J. Sadowsky, Andrew M. Odlyzko, Jeffrey A. Gralnick, and Igor G. L. Libourel. Genome-Scale Metabolic Network Validation of Shewanella oneidensis Using Transposon Insertion Frequency Analysis. *PLoS Comput Biol*, 10(9):e1003848, sep 2014. doi: 10.1371/journal.pcbi.1003848.

Christopher M. Yates, Ioannis Filippis, Lawrence A. Kelley, and Michael J. E. Sternberg. SuSPect: Enhanced Prediction of Single Amino Acid Variant (SAV) Phenotype Using Network Features. *J Mol Biol*, 426(14):2692–2701, jul 2014. doi: 10.1016/j.jmb.2014.04.026.

Z.-Q. Ye, S.-Q. Zhao, G. Gao, X.-Q. Liu, R. E. Langlois, H. Lu, and L. Wei. Finding new structural and sequence attributes to predict possible disease association of single amino acid polymorphism (SAP). *Bioinformatics*, 23(12):1444–1450, jun 2007. doi: 10.1093/bioinformatics/btm119.

Peng Yue, Eugene Melamud, and John Moult. SNPs3D: Candidate gene and SNP selection for association studies. *BMC Bioinformatics*, 7(1):166, 2006. doi: 10.1186/1471-2105-7-166.

Nan Zhao, Jing Ginger Han, Chi-Ren Shyu, and Dmitry Korkin. Determining Effects of Non-synonymous SNPs on Protein-Protein Interactions using Supervised and Semi-supervised Learning. *PLoS Comput Biol*, 10(5):e1003592, may 2014. doi: 10.1371/journal.pcbi.1003592.

I have not failed. I've just found 10,000 ways that won't work.

Thomas Edison

6

Predicting enzyme kinetics: the quest for the holy grail

Summary

This chapter describes the landscape of catalytic rates across multiple species and reactions. The challenges and approaches to bridge data from different databases, including enzyme sequences and structures as well as chemical compounds present in the reactions are also explained in this chapter. Overall, here the reader can find how different aspects of biology and chemistry are related to the enzyme activities.

Abstract

Mutagenesis and evolution-based strategies has been used to develop strains for industrial applications for decades, but these strategies usually create strains with many mutations. Due to developments in sequencing technology, we can now routinely resequence strains to determine the mutations that are present. However, we lack tools to understand the effect of most of these mutations without testing them in the laboratory one at a time. As described in previous chapter, mutations can affect metabolic functions in multiple ways including modulation of gene expression through intragenic mutations and mutations in actual enzyme coding sequences. In this chapter we focus on mutations in enzyme coding sequences that could affect enzyme activity through influencing the stability of enzymes, their catalytic rates (k_{cat}), or their affinity to substrates. Here we focus primarily on exploring the mutational and other determinants of enzyme k_{cat} values using publicly available in vitro k_{cat} data. We collect the data for this analysis by merging enzyme kinetics data with protein sequence and structure as well as metabolite structure information from multiple databases. We use the dataset to study how predictable k_{cat} values are from features related to enzyme sequences, enzyme structures as well as features of chemical substrates and products of the reaction. Due to incompleteness of datasets as well as lack of cross references between databases, the full dataset has only limited coverage of the enzyme universe. We can still use this limited dataset to explore determinants of k_{cat} values at least for a subset of enzymes. We observe no direct correlation between predicted mutation effect of specific mutations and catalytic rates for the small number of enzymes where this data is available. However, we show that certain mutations that are predicted to destabilize the protein can result in reduced k_{cat} values. Finally, we build a statistical model to predict k_{cat} values from all the protein sequence/structure and metabolite/reaction features. This model shows that certain features of the chemical transformations catalyzed by enzymes can be used to predict the catalytic rate. In conclusion, we find that at the present moment it is not possible to develop general models that explain how mutations affect catalytic activities of enzymes primarily due to lack of comprehensive datasets that span enough variants of different enzymes. However, it is possible to use the dataset that we have created to study general determinants of catalytic rates.

Introduction

Genetic variation is the motor of evolution. Organisms change their genotype to adapt to changing conditions in the environment. Individuals with favorable mutations will prevail, leading to the survival of the population. We can harness this characteristic using adaptive laboratory evolution (ALE). ALE consists of selecting the fittest individuals in a population during the course of time, while they are exposed to stress conditions. For example, selecting the fastest growing strains under carbon limitation, high-temperature or presence of toxic chemicals (Hansen et al., 2017).

The mutant strains obtained from ALE experiments contain two types of genetic variations. The first type confers a new phenotype and allow the new individuals to survive. The impact of these mutations in the protein activities can be difficult to predict, except if it is a clear disruption of the protein (e.g., early stop codon). Other mutations are neutral (i.e., have no measurable impact on the phenotype) and occur by chance. Predicting the effects of mutations in individual enzymes can help distinguish causal from neutral mutations.

Genome-scale metabolic models (GEMs) describe the reactions occurring inside a cell and their relation to the phenotype, encoded by gene-protein-reaction (GPR). These models have found applications across multiple fields, including metabolic engineering, cancer analysis, phylogenetic analysis, etc (O'Brien et al., 2015). In a metabolic model, each reaction can occur up to a maximum rate v_{max}. The v_{max} is determined by the enzyme concentration $[E]$ and the turnover number k_{cat} ($v_{max} = [E] \cdot k_{cat}$). Therefore, these parameters can be used to constraint the model, by limiting the maximum flux of each individual reaction. Incorporating enzyme concentrations and turnover numbers has proven to improve phenotype predictions in *Saccharomyces cerevisiae* using a GEM (Sánchez et al., 2017).

Incorporating the impact of genetic variation in metabolic model would result in better phenotype predictions and could lead to a better understanding the causality of mutations occurring in enzymes. To accomplish that, we need to calculate the change in k_{cat} and concentration for a set of given enzymes with mutations and use that information to constraint the flux limits. Using these values we can directly assess the effect of mutations in the phenotype (Jamshidi et al., 2007). However, we lack of predictive models capable of predicting the change in k_{cat} or concentrations given a change in DNA sequence.

As described in the previous chapter, many tools have been published in the medical field to assess the effect of mutations in diseases, with a high focus in cancer research. Mutations leading to cancer or other health disturbances do not necessarily correlate with increase or decrease of a given protein activity and therefore cannot be used to explain the impact of genetic variation on a single protein (Cardoso et al., 2015). It is possible to predict enzyme kinetics using machine-learning approaches for single enzymes. That, however, requires enough examples of reactions or mutants and good quality protein structures (Carlin et al., 2016, Sipilä and Taskinen, 2004).

In this work we explored the global landscape of k_{cat} values across multiple types of enzymes using the curated kinetics data available at SABIO-RK (Wittig et al., 2012), protein sequence and structure databases, and reaction substrate/product databases. Ideally, we would build structure-based models for enzyme catalytic rates for each enzyme that would allow predicting the effect of mutations. Indeed, we have structural models for almost every enzyme in *Escherichia. coli* (Brunk et al., 2016). Still, we do not have structural models for many other organisms and in particular we do not have systematic measurements k_{cat}s for a reasonable number sequence variants of a single enzyme. Therefore, we cannot build protein specific predictive models whenever we need. Instead, we decided to build a general model with the examples available from multiple proteins across different species.

We first analyzed the relationship between predictect mutation effects k_{cat} changes for the small number of enzymes in SABIO-RK where sufficiently large number of variant sequences were available. We predicted mutation effects based on sequence alone using Sorting Intolerant from Tolerant (SIFT) (Ng and Henikoff, 2001) and based on structural data using foldX (Guerois et al., 2002). These analyses showed that mutation effects predicted using sequence conservation (SIFT) do not predict changes in k_{cat} at all. Structural approaches (foldX) had some predictive power, but still there was limited correlation between predicted mutation effects and k_{cat} changes.

In order to study more generally the determinants of catalytic activity beyond mutations, we computed 284 features using the data available (187 reaction specific features related to the substrates and products of a reaction, 95 protein specific features related to both sequence and structural features, pH and temperature of the enzyme kinetic experiment) and used them to train different regression algorithms to predict k_{cat}s. To generate reaction and enzyme features, we needed to integrate data from multiple databases (KEGG, ChEBI, PubChem, UniProt and PDB). In every part of the process, we lost parts of the data due to lack or inconsistent identifiers

in the SABIO-RK database and incorrect locations of mutations reflecting the lack of proper identifiers in the biochemistry literature. The final complete data set is a under representation of all the enzyme kinetics data available (2%) and it is not likely to represent well the whole universe of enzymatic functions. However, it is possible to use the data and regression models trained on the data to explore determinants of the variation in k_{cat} values in the data.

We highlight interesting facts about the k_{cat} landscape. First, evolutionary conservation alone does not seem to explain the catalytic rates on its own. Second, the effects of protein stability must be accounted for in using measured $k_{cat}s$. Finally, the maximum catalytic rates observed vary with EC number and the actual chemistry that takes place. That suggests that the chemistry of the reaction could limit the space of possible kinetic values. However, the amount of data is not currently sufficient to do deeper assessment of general factors driving variation in catalytic activity.

Materials and Methods

Retrieving SABIO-RK data

Kinetic data was retrieved from SABIO-RK (Wittig et al., 2012). The workflow to retrieve SABIO-RK data can be found in Figure 6.1 The data was retrieved using their Web API. We collected the following information: reaction stoichiometry, metabolite cross-references, enzyme functions and cross-references, and kinetic parameters (i.e., k_{cat}, k_M, Hills constants, temperature and pH). During the processing step, we discard polymerization reactions.

Enrich metabolites identifiers

Using the Chemical Translation Service (CTM) to enrich our metabolites database (Wohlgemuth et al., 2010). For every metabolite in our database without a reference, we queried the CTM using metabolite names and accepted only entries that were 100% match. Using the data from CTM we added cross-references (if available) for the following databases: KEGG, ChEBI and PubChem.

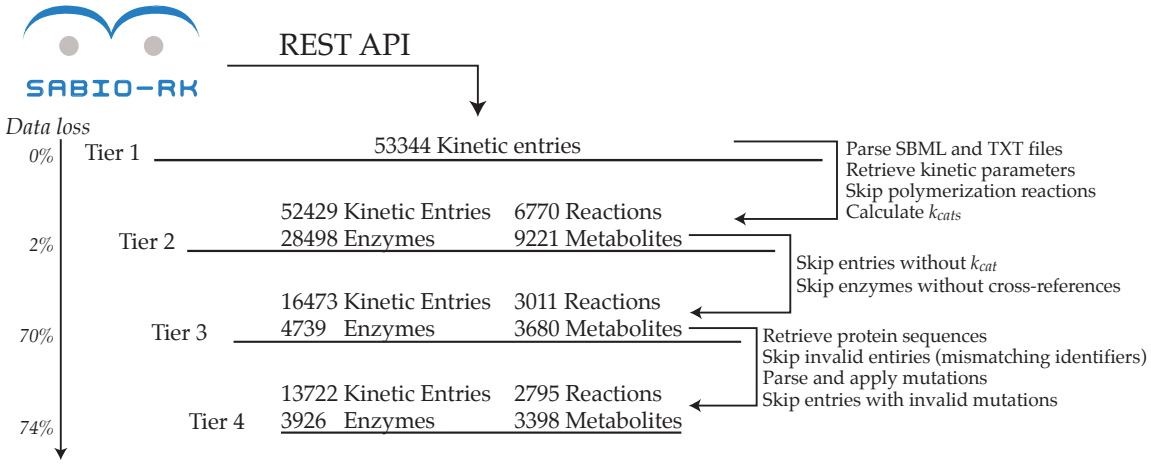


Figure 6.1: Worflow for k_{cat} retrieval. Using the REST API we download 53344 SBML files describing each a single kinetic experiment. Every tier represents the data after a processing step. The amount of data left is reported on the left.

Clustering the experiments by sequence

We retrieved the peptide sequences of each enzyme in our data set from UniProt. We generated mutant specific sequences by applying the mutations described in the annotation. To build the clusters we used UCLUST (Edgar, 2010). The clusters were computed using the following parameters: $id = 0.6$, $target\_cov = 0.7$, $query\_cov = 0.7$ and $minqt = 0.5$. All sequences were sorted in descending order based on their length as suggested in UCLUST documentation. We evaluated the clustering results by measuring the information entropy of the EC number annotations (using the first 2 digits). The information entropy was calculated using the following equation:

$$- \sum_{i=0}^{n} \cdot P(x_i) \cdot log_{10}\, P(x_i)$$

where n is the number of observations and $P(x_i)$ is the probability of observing x_i in the data.

Correcting inconsistent data entries

When multiple k_{cat}s were available for the same reaction catalyzed at the same pH and temperature by the same enzyme (with specific mutations), we selected the highest measured k_{cat} value.

Correcting mutation positions

We tested the SNVs, deletions, and indels against the sequences from UniProt. In some cases, the mutated amino-acid does not match the sequence. We assume this is a consequence of peptide sequencing used before NGS. To overcome that issue, we test the mutations against post-processed sequences to identify corrected position of each mutation, using the post-translational modifications (PTMs) and other processing events identified at UniProt. However, we can only validate deletions, SNVs and indels are because the original amino-acid is reported in the mutation notation. Insertion mutations cannot be mapped because they contain only the insertion position. So, we only corrected the insertion mutations when there occurred together with other mutations.

Conservation scores

We determined the sequence conservation in 6 steps:

1. Group the sequences by reaction, sequence cluster, pH and temperature.
2. Select the longest available sequence in each group to be the reference.
3. Calculate the mutations required to convert a given sequence in the group into the reference sequence. The mutations were identified from a global sequence alignment. We used the BLOSUM62 scoring matrix, a gap open penalty of -10 and a gap extension penalty of -10 for the alignment.
4. Using SIFT software, we built a score matrix for every amino-acid in all positions. We assigned the gap score at a given position to the average score of all amino-acids at that position.
5. For every sequence in a group, we applied all the necessary gaps to match the reference sequence. The modified sequence should have the same length as the reference sequence.
6. For every position in a sequence and the reference sequence, we retrieved the tolerance score of each amino-acid or gap at the given position. The sequence conservation score is the sum of all positional scores.

   The change in conservation score is calculated using the following function:

$$\Delta C_{score} = |C_{ref} - C_{other}|/C_{length},$$

where the C_{ref} is the conservation score of the reference sequence, C_{other} is the conservation score of the sequence we are comparing and S_{length} is the size of the largest sequence.

UniProt and PDB bridging pipeline

We created a workflow to convert the protein structures found in PDB into structures representing the wild-type sequences from UniProt. This pipeline is necessary to remove cloning tags and mismatching amino-acids between UniProt and PDB records.

The workflow consists of the following steps:

1. We retrieved all PDB structures identified in the UniProt cross-reference section. The peptide chains in the PDB file and their relative positions after post-translational modification are also reported in the UniProt data.

2. For each peptide chain in the PDB file, we matched a peptide chain reported in the UniProt sequence. We did that by matching peptide chain positions described in the UniProt records if more than one chain is reported.

3. We align the peptide chains to identify small modifications. We applied the global sequence alignment algorithm, using the BLOSUM62 scoring matrix.

4. If the mutations resulting from the alignment were SNVs or deletions, we created a new protein structure. We removed the amino-acids reported as deletions and replaced of SNVs using the *ssbio* library (Mih et al., 2017). Structures requiring insertion mutations were discarded at this step.

5. Finally we used the foldX repair function on the modified files.

Mutation effects related to protein stability

We used foldX to compute the stability ΔG. The foldX Stability command takes the temperature and pH as parameters. We built structural models with the specific mutations and calculated the ΔG of each condition. The $\Delta\Delta G$s were calculated by subtracting the ΔGs of individual simulations.

Feature generation

We used three kinds of features: protein features, reaction features and process features.

Protein features. We computed 95 features using the protein sequences and strucutres. The feature set includes the frequency of each amino-acid, the frequency of charged, polar and basic amino-acids, the grand average of hydropathy (GRAVY), the ΔG energy terms of folding predicted by foldX, EC numbers (up to the third digit), among others. The complete list of features can be found in Table S6.1.

Reaction features. Using ChemAxon Calculator Plugins `http://www.chemaxon.com` and RDKit (`http://www.rdkit.org`), we computed 187 features for each reaction. Reaction features are computed using the metabolites (i.e., substrates features, product features) and the conversion (diff between substrate and product features). The feature set includes 75 functional groups, the number of bonds for (including bond types), polarizability, polar surface area (PSA), non-polar surface area (NPSA), and charge, among others. The complete list of reaction features can be found in Table S6.2.

Machine Learning

We trained four linear regression models using the features described before to predict the k_{cat}s. We used a linear model fitted using LASSO, ridge regression, LARS, and LASSO-LARS learning algorithms. The models, training methods and cross-validation functions are implemented in the *scikit-learn* package (Pedregosa et al., 2011).

PCA analysis of reactions for each EC number

We split the data by EC number and performed principal component analysis (PCA) using the reaction features. We used the highest k_{cat} for each reaction to color points in the scatter plots.

Results and Discussion

Available K_{CAT} data

SABIO-RK contains 53344 entries representing experiments to determine kinetic parameters (e.g., turnover numbers, Michaelis constants, inhibition constants, maximal velocity, etc.). Only 16473 entries contain k_{cat} measurements and a valid sequence cross-reference to UniProt.

The enzymes retrieved from SABIO-RK are described by name, UniProt reference (if available) and mutations. To generate non-redundant identifiers, we used the UniProt references and mutations. When no UniProt references were available, we generated a unique identifier for the enzyme. We identified 4739 unique enzymes with cross-references (Figure 6.1). We then applied the position correction algorithm to match the mutations to UniProt sequences. The final number of enzymes (wild-type and valid mutants) is 3926 (Figure 6.1).

We also noticed that SABIO-RK entries had multiple k_{cat}s measurements for the same experimental setup (i.e., same enzyme, reaction, pH and temperature). We merged 3058 redundant experimental setups into unique records by keeping the highest k_{cat} for each repeated entry. Our final data set contains 11288 entries each with a unique combination of enzyme, reaction, pH and temperature.

We split the data into three subsets using clustering. The "all-measurements", containing all the 11288 entries that we could retrieve from SABIO-RK; "ecoli-measurements", containing the subset of data where enzymes from *E. coli* and close related species can be found; and "yeast-measurements", containing the subset of data for *Saccharomyces cerevisiae* and other close yeast strain enzymes.

Clusters containing sequences belonging to *E. coli* strains, the "ecoli-measurements" set, contains 1192 entries. This corresponds to 154 distinct clusters of similar sequences and 394 unique reactions. The data set containing yeast strains, the "yeast-measurements" data set, contains 493

entries, corresponding to 160 reactions and 62 sequence clusters.

Comparison between K_{CAT} and protein sequences

We computed the conservation scores for every peptide sequence relative to a reference sequence in each sequence cluster in the "ecoli-measurements" and "yeast-measurements" subsets. Because temperature and pH have an effect on the kinetic parameters, we grouped our data by sequence cluster, reaction, pH and temperature. We observed no correlation between the variation in sequence conservation and the variation k_{cat} in both yeast and *E. coli* data sets (Figure 6.2). Indeed, the sequence conservation based on position specific probabilities fails to capture inter-residue interactions (Hopf et al., 2015). We need a more comprehensive model to link amino-acid sequences with enzymatic activity. However, to build such model, we need to collect more data.

Comparison between K_{CAT} and folding stability

We used foldX to compute the ΔG of stability for each protein in the "ecoli-measurements" set. The stability ΔG accounts for pH and temperature. For each cluster, we compared the $\Delta\Delta G$ of stability between different protein sequences with the change in k_{cat} within each protein cluster for different reactions. We applied our UniProt-PDB bridging pipeline on the *E. coli* data set to create structures representing the wild-type and mutant sequences. There are two main reasons why we changed the files PDB files: the proteins expressed for crystallization contain expression tags and the sequences reported differ from the wild-type. We manage to recover 50 structures from the 204 distinct protein sequences available in our data.

We split the data by cluster and reaction, because foldX calculations account for temperature and pH. We plotted the $k_{cat}s$ and ΔGs to identify correlation patterns and found two groups where between stability and k_{cat} are correlated (Figure 6.3).

In the first group, the kinetic parameters were measured at $30\,\mathrm{deg}$ Celsius (Table 6.1). The pH was 6.9 for two of the three experiments. This suggests that the k_{cat} is influences by protein stability in this cluster. Still, with only three points this effect should not be considered completely explanatory.

The second group contains also three measurements at constant pH and temperature (7.5, $22.0\,\mathrm{deg}$ Celsius). This indicates that the mutations make the protein more unstable and there-

Table 6.1: First group description.

Temperature (°C)	pH	k_{cat}	UniProt Id	ΔG	Mutations
30.0	6.9	0.95	P37610	-50.95	
30.0	6.9	1.25	P37610	-51.53	S158A
30.0	8.0	1.83	P37610	-52.82	

Table 6.2: Second group description.

Temperature (°C)	pH	k_{cat}	UniProt Id	ΔG	Mutations
22.0	7.5	0.002	P0AES2	-211.75	K207R
22.0	7.5	0.280	P0AES2	-213.16	N341D
22.0	7.5	44.000	P0AES2	-205.75	

fore the k_{cat} decrease might be strongly influenced by the amount of active enzyme (Table 6.2).

Fitting the data using linear regression

We generated 284 potentially predictive features for the "all-measurements" data set. The feature generation approach we applied is very strict and requires identifiers all chemicals and enzymes (including 3D structures), therefore the final data set comprises only 1012 entries (Table 6.3). The data loss has 4 different causes: inconsistent UniProt identifiers, lack or inconsistent metabolite identifiers, invalid mutation mapping, and lack of 3D structures.

We evaluated the distribution of EC numbers in the data set. We observed that Ligases and Oxireductases were underrepresented in the data (Figure 6.4). At this point, we did not try to balance the enzyme classes because we a limited amount of data.

Table 6.3: Overview of the data used to build linear regression models.

Feature	Count
Entries	1012
EC numbers	98
Species	59
Features	284
Wild-Type Enzymes	127
Mutant Enzymes	244

Table 6.4: Residual sum of squares for different linear regression models.

Data Set	lasso	ridge	lars	lasso-lars
Train	$7.371E3$	$4.785E3$	$1.052E4$	$8.635E3$
Test	$1.795E3$	$1.561E3$	$2.329E3$	$1.946E3$

Table 6.5: Feature count for different algorithms

Algorithm	Feature count
LASSO	43
RIDGE	283
LARS	11
LASSO-LARS	23

We applied four different learning algorithms to build linear models: LASSO, ridge regression, LARS, and LASSO-LARS. Using the traditional machine learning approach, we split our data into train and test (80% and 20%) and fitted the model parameters using cross validation. Table 6.4 shows the residual sum of sequares (RSS) for each model and predictions are shown in Figure 6.5.

Given the high number of features and the low amount of data, the fitted models is likely to be overfitted. Plus, the 1012 data points represent 2% of the original entries in SABIO-RK indicating severe undersampling of the enzyme space. Therefore, this model will not necessarily generalize beyond the dataset used in the present study.

We selected the most predictive features (top 20 if available, Figures S6.5, S6.6, S6.7 and S6.8). Despite the fact that we cannot use any of the models to predict k_{cat}s in general, we can still look at patters in the data. The feature selection identified features corresponding to chemical changes as a result of fitting the data. Applying LASSO, LARS and LASSO-LARS result in higher sparsity in the feature selection, while ensuring a reasonable fit which allows better interpretability (Table 6.5). The selected features indicate that the nature of the chemical reaction contributes to determine k_{cat}.

Relationship between catalytic rates and with the chemistry of the reactions

To identify if there is any relationship between the chemistry (i.e., the process of chemical conversion) and the catalytic rates, grouped our data by EC number. Because we do not need to know the UniProt or chemical identifiers, we can use all entries with k_{cat} (23293 measurements).

Enzymes with more than one EC number were classified as complex (Figures S6.9, S6.10, S6.11 and S6.12). There are some distinct patterns. For example EC numbers 1.14.15, 1.14.17, 1.13.99 or EC 4.99.1 tend to display a slower rate. Other enzymes, with EC numbers 1,3.99 or 1.10.99 show a higher $k_{cat}s$.

We hypothesize that some catalytic rates might be limited by the chemistry of the reaction they catalyze. It is possible to predict the impact of mutations in a single protein using machine-learning (Carlin et al., 2016). Interestingly, the study by Carlin et al. Carlin et al. (2016) also found that the hydrogen bonding energy of the substrate was a predictive feature.

With that in mind, we calculated Pearson correlation coefficients between EC numbers and product features and EC numbers and substrate features. We observed that a some features correlate strongly with certain EC numbers. We also observed a low to moderate correlations between the substrate or product features and most of the EC numbers (Figures S6.13 and S6.14). The top 10 EC numbers with the highest correlation coefficients with substrate or products features are show in Table 6.6.

Using PCA we decomposed the subset of data corresponding to each EC number, using the reaction features. We plotted the transformed data using the first two components and colored the points with k_{cat} values. In a few cases, we observed a good separation between high and low $k_{cat}s$ based on the PCA decomposition (Figures S6.15, S6.16 and S6.17).

Conclusions

The major challenge with the present study was creating a consistent dataset that links enzyme kinetic parameters with sequence and structural information. The kinetic data available in SABIO-RK does not have consistent identifiers. The database lacks protein identifiers for approximately half of the enzymes described in the entries. One reason for that is the lack of standards for pub-

Table 6.6: Top EC numbers and correlated features ranked by Pearson correlation coefficient. The pairs with correlation coefficient higher than 0.75 are shown.

EC	Coefficient	Feature
1.7.3	1.00	Products number of nitroso
1.1.3	1.00	Substrates number of peroxyde
3.4.21	0.96	Substrates number of generic amino acid (not glycine)
3.4.21	0.94	Products number of generic amino acid (not glycine)
1.11.1	0.91	Products number of peroxyde
3.1.1	0.88	Products number of anhydrides (except formic anhydride)
6.4.1	0.88	Substrates number of carbamate
6.3.4	0.88	Substrates number of carbamate
1.1.1	0.84	Products number of enanmine
5.3.1	0.81	Products number of aldehyde
4.1.2	0.76	Substrates number of aldehyde

lishing enzyme kinetics until the early 80's. Still, despite the adoption of standards in the 90's, most of the articles on enzyme kinetics do not report EC numbers nor protein and reaction identifiers (Wittig et al., 2014). In some cases, SABIO-RK entries were reassigned in UniProt and the identifiers were no longer valid.

Recent work has shown that machine-learning can be used to predict enzyme kinetic parameters of a single enzyme from sequence variant data using a training set of 100 mutants (Carlin et al., 2016). To make a model capable of predicting k_{cat}s in general from sequence variation data requires similar datasets for a large number of enzymes. Our data set collected from SABIO-RK represents approximately 2% of the existing k_{cat}s. measurements and does not contain enough cases where both wild type and mutant versions of the same enzyme have consistently measured parameters. This limits our current ability to explore general approaches for predicting the effects of sequence variation on enzyme kinetics.

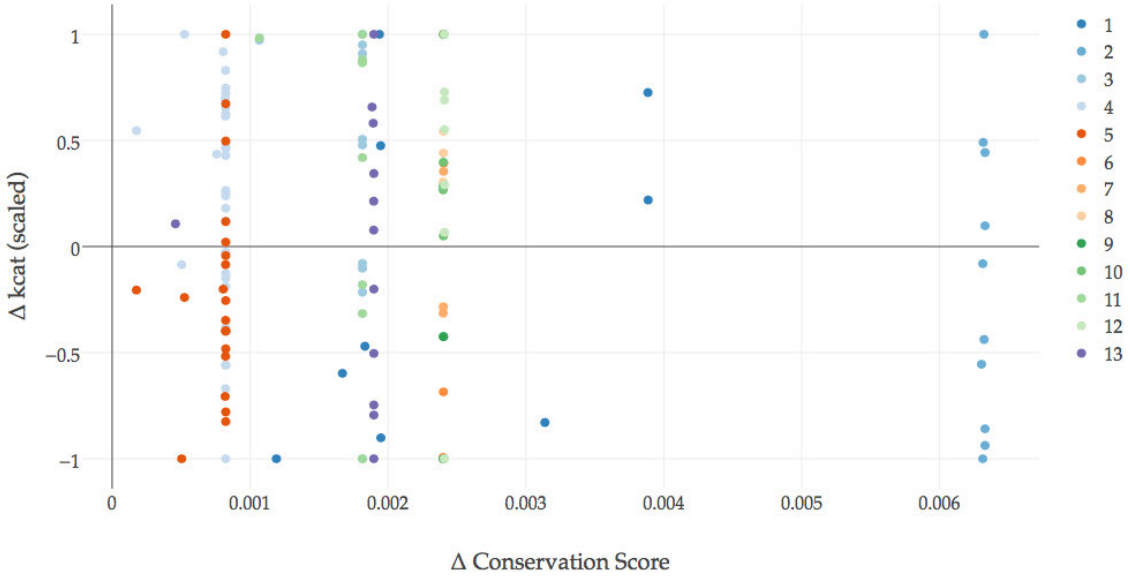
Deep mutational scanning (DMS) provides a systematic way to create a sequence-to-fitness landscapes for enzymes whose activity can be measured by a high throughput assay (e.g. FACS) or connected to cellular fitness (Fowler and Fields, 2014). This type of data sets could be used to learn more generally how sequence variation affects enzyme functions. Data from DMS experiments is accumulating in the public domain, but these experiments have still only been performed for a handful of enzymes. In addition the sequence diversity that can be spanned in DMS exper-

iments is still a small fraction of the full sequence landscale. For example, a protein with 200 amino-acids has $1.048576E^{46}$ possible distinct sequences, so we would need to cherry-pick parts of the proteins to study in more detail. In conclusion, our ability to quantitatively predict effects of sequence variation on enzyme kinetics is primarily limited by availability of relevant data from consistently performed experiments. This also limits our ability to predict more complex genotype-phenotype relationships such as how mutations discovered by resequencing ALE isolates affect metabolic phenotypes.

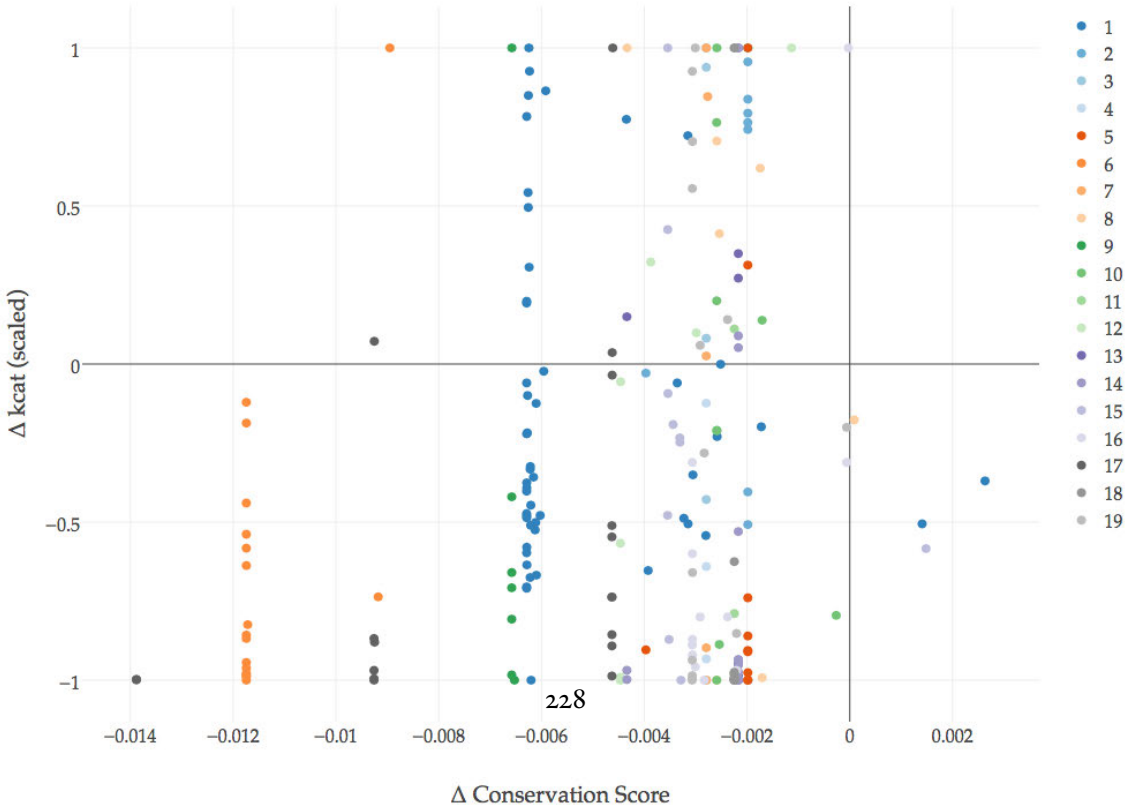
Due to limitations in the data available to explore the enzyme sequence-k_{cat} relationship, we decided to instead focus on more broadly studying the determinants of k_{cat} values including features of the chemical reaction that is taking place as well as sequence and structure features. Previous studies show that the measurements of kinetic constants *in vitro* are close to *in vivo* expected value for *E. coli* based on estimated metabolic fluxes and measurements of absolute protein levels (Davidi et al., 2016). This could mean that evolution of has selected for the specific catalytic efficiencies we observe, i.e. that the k_{cat} values are not intrinsically limited by factors related to reaction chemistry. We show that we can split some of our data into high and low k_{cat}s using only information about the chemistry,. Moreover, the data we obtained shows that the catalytic rate of some reactions depends on the chemistry itself. This hypothesis needs to be further investigated because this observation can be a result of lack of available kinetic measurements for different groups of reactions.

Three questions arise from these observations. Can we identify physicochemical constraints that limit the kinetics of biochemical reaction? Can we quantify complex interactions between enzymes and reactions using machine-learning without structural modeling? Did we observe enough reactions to identify the limits imposed by chemistry?

To address this questions we to solve three problems. First, we need more data, to make sure we don't collect artifacts from the data analysis itself. Second, we need to separate the reactions using an unbiased method. While the first EC number separates the reaction by mechanism, the nomenclature is artificial, incomplete and some times not consistent. Finally, we need to learn more about the interaction of substrates and enzymes: what kind of bonds do they form and what are the specific steps of the conversion.



(a) *E. coli* clusters



(b) Yeast clusters

Figure 6.2: Variation in protein sequence conservation and k_{cat}. The Δk_{cat} values were scaled between -1 and 1 for plotting.

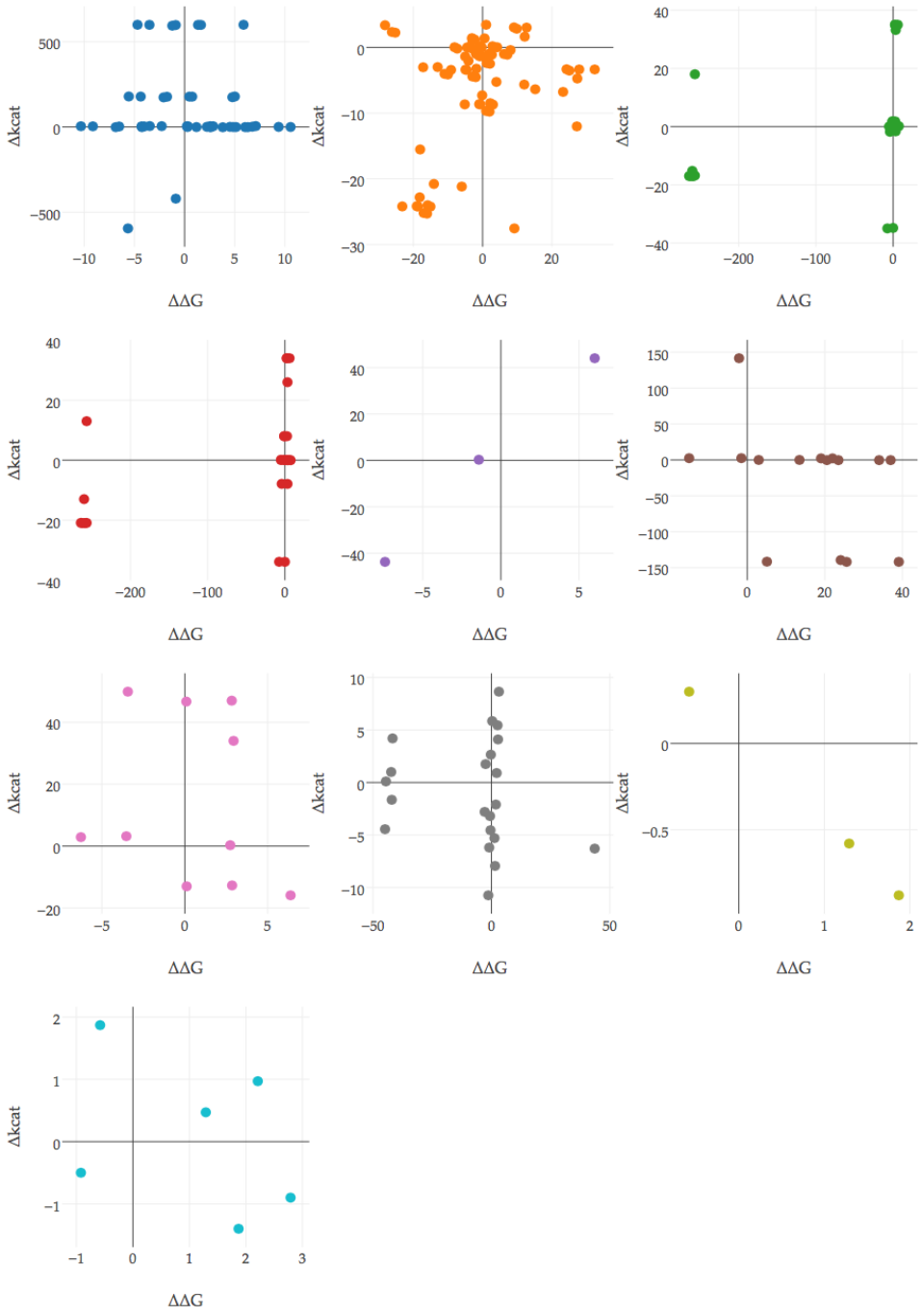


Figure 6.3: Change in stability and k_{cat} within protein clusters and reactions. Each subplot shows the $\Delta\Delta G$ and change in k_{cat} between each element of the cluster against all other elements in the same cluster that catalyze the same reaction.

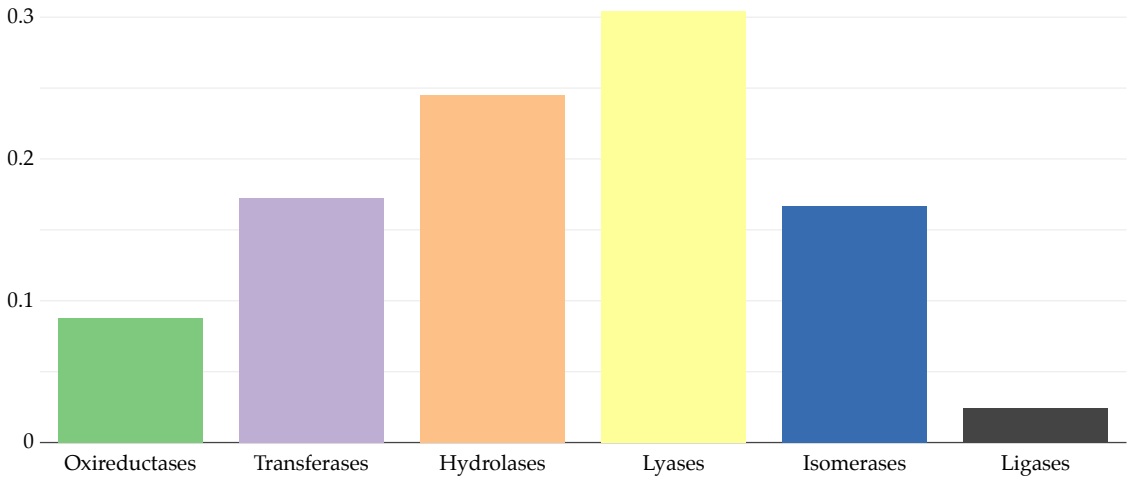


Figure 6.4: Distribution of enzymes per EC group. EC 1: Oxireductases; EC 2: Transferases; EC 3: Hydrolases; EC 4: Lyases; EC 5 Isomerases; and EC 6 Ligases

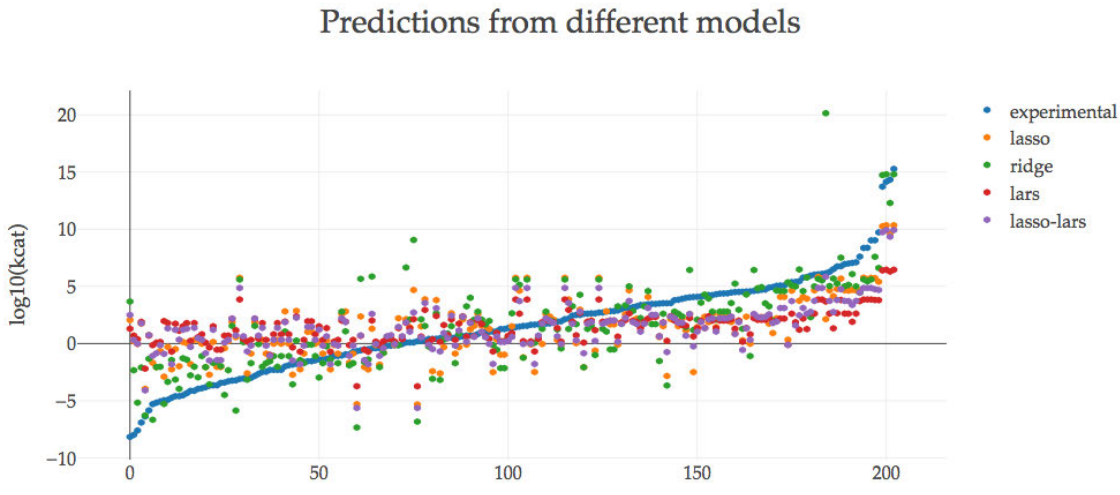


Figure 6.5: Experimental and predicted $log_{10}(k_{cat})$. This scatter plot shows the experimental and predicted values. The data was sorted by experimental value in ascending order.

References

Elizabeth Brunk, Nathan Mih, Jonathan Monk, Zhen Zhang, Edward J. O'Brien, Spencer E. Bliven, Ke Chen, Roger L. Chang, Philip E. Bourne, and Bernhard O. Palsson. Systems biology of the structural proteome. *BMC Systems Biology*, 10(1):26, 2016. ISSN 1752-0509. doi: 10.1186/s12918-016-0271-6.

João G. R. Cardoso, Mikael Rørdam Andersen, Markus J. Herrgård, and Nikolaus Sonnenschein. Analysis of genetic variation and potential applications in genome-scale metabolic modeling. *Frontiers in Bioengineering and Biotechnology*, 3(13):1–12, 2015. ISSN 22964185. doi: 10.3389/fbioe.2015.00013.

Dylan Alexander Carlin, Ryan W. Caster, Xiaokang Wang, Stephanie A. Betzenderfer, Claire X. Chen, Veasna M. Duong, Carolina V. Ryklansky, Alp Alpekin, Nathan Beaumont, Harshul Kapoor, Nicole Kim, Hosna Mohabbot, Boyu Pang, Rachel Teel, Lillian Whithaus, Ilias Tagkopoulos, and Justin B. Siegel. Kinetic characterization of 100 glycoside hydrolase mutants enables the discovery of structural features correlated with kinetic constants. *PLoS ONE*, 11 (1), 2016. ISSN 19326203. doi: 10.1371/journal.pone.0147596.

Dan Davidi, Elad Noor, Wolfram Liebermeister, Arren Bar-Even, Avi Flamholz, Katja Tummler, Uri Barenholz, Miki Goldenfeld, Tomer Shlomi, and Ron Milo. Global characterization of in vivo enzyme catalytic rates and their correspondence to in vitro k _{cat} measurements. *Proceedings of the National Academy of Sciences*, 113(12):3401–3406, 2016. ISSN 0027-8424. doi: 10.1073/pnas.1514240113.

Robert C. Edgar. Search and clustering orders of magnitude faster than BLAST. *Bioinformatics*, 26(19):2460–2461, oct 2010. ISSN 1460-2059. doi: 10.1093/bioinformatics/btq461.

Douglas M Fowler and Stanley Fields. Deep mutational scanning: a new style of protein science. *Nature Methods*, 11(8):801–807, 2014. doi: 10.1038/NMETH.3027.

Raphael Guerois, Jens Erik Nielsen, and Luis Serrano. Predicting Changes in the Stability of Proteins and Protein Complexes: A Study of More Than 1000 Mutations. *Journal of Molecular Biology*, 320(2):369–387, jul 2002. ISSN 00222836. doi: 10.1016/S0022-2836(02)00442-4.

Anne Sofie Lærke Hansen, Rebecca M. Lennen, Nikolaus Sonnenschein, and Markus J Herrgård. Systems biology solutions for biochemical production challenges. *Current opinion in biotechnology*, 45:85–91, jun 2017. ISSN 1879-0429. doi: 10.1016/j.copbio.2016.11.018.

Thomas A. Hopf, John B. Ingraham, Frank J. Poelwijk, Michael Springer, Chris Sander, and Debora S. Marks. Quantification of the effect of mutations using a global probability model of natural sequence variation. *Nature Publishing Group*, 35(2), 2015. ISSN 1087-0156. doi: 10.1038/nbt.3769.

Neema Jamshidi, Thuy D Vo, and Bernhard O Palsson. In silico analysis of SNPs and other high-throughput data. *Methods in molecular biology (Clifton, N.J.)*, 366:267–85, jan 2007. ISSN 1064-3745. doi: 10.1007/978-1-59745-030-0_15.

Nathan Mih, Elizabeth Brunk, Ke Chen, Edward Catoiu, Anand Sastry, Erol Kavvas, Jonathan M Monk, Zhen Zhang, and Bernhard O Palsson. ssbio : A Python Framework for Structural Systems Biology. *bioRxiv*, 2017. doi: 10.1101/165506.

Pauline C Ng and Steven Henikoff. Predicting deleterious amino acid substitutions. *Genome research*, 11(5):863–74, may 2001. ISSN 1088-9051. doi: 10.1101/gr.176601.

Edward J. O'Brien, Jonathan M. Monk, and Bernhard O. Palsson. Using genome-scale models to predict biological capabilities. *Cell*, 161(5):971–987, 2015. ISSN 10974172. doi: 10.1016/j.cell.2015.05.019.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Benjamín J Sánchez, Cheng Zhang, Avlant Nilsson, Petri-Jaan Lahtvee, Eduard J. Kerkhoven, and Jens Nielsen. Improving the phenotype predictions of a yeast genome-scale metabolic model

by incorporating enzymatic constraints. *Molecular Systems Biology*, 13(8):935, aug 2017. ISSN 1744-4292. doi: 10.15252/msb.20167411.

Julius Sipilä and Jyrki Taskinen. CoMFA Modeling of Human Catechol O-Methyltransferase Enzyme Kinetics. *Journal of Chemical Information and Computer Sciences*, 44(1):97–104, 2004. ISSN 00952338. doi: 10.1021/ci034189k.

Ulrike Wittig, Renate Kania, Martin Golebiewski, Maja Rey, Lei Shi, Lenneke Jong, Enkhjargal Algaa, Andreas Weidemann, Heidrun Sauer-Danzwith, Saqib Mir, Olga Krebs, Meik Bittkowski, Elina Wetsch, Isabel Rojas, and Wolfgang Müller. SABIO-RK - Database for biochemical reaction kinetics. *Nucleic Acids Research*, 40(D1):790–796, 2012. ISSN 03051048. doi: 10.1093/nar/gkr1046.

Ulrike Wittig, Renate Kania, Meik Bittkowski, Elina Wetsch, Lei Shi, Lenneke Jong, Martin Golebiewski, Maja Rey, Andreas Weidemann, Isabel Rojas, and Wolfgang Müller. Data extraction for the reaction kinetics database SABIO-RK. *Perspectives in Science*, 1(1-6):33–40, 2014. ISSN 22130209. doi: 10.1016/j.pisc.2014.02.004.

Gert Wohlgemuth, Pradeep Kumar Haldiya, Egon Willighagen, Tobias Kind, and Oliver Fiehn. The chemical translation service-a web-based tool to improve standardization of metabolomic reports. *Bioinformatics*, 26(20):2647–2648, 2010. ISSN 13674803. doi: 10.1093/bioinformatics/btq476.

Supplementary Information

Table S6.1: Feature weight for protein features using different machine-learning models. We used for regression methods: LASSO, ridge, LARS, LASSO-LARS

feature	LASSO	RIDGE	LARS	LASSO-LARS
EC 1.1.1 Oxidoreductases acting on the ch-oh group of donors with nad(+) or nadp(+) as acceptor	-0.029	-2.646	0.000	0.000
EC 1.1.3 Oxidoreductases acting on the ch-oh group of donors with oxygen as acceptor	0.000	1.728	0.000	0.000
EC 1.11.1 Oxidoreductases acting on a peroxide as acceptor peroxidases	1.158	-0.248	0.000	0.585
EC 1.3.5 Oxidoreductases acting on the ch-ch group of donors with a quinone or related compound as acceptor	0.000	0.394	0.000	0.000
EC 1.3.8 Oxidoreductases acting on the ch-ch group of donors with a flavin as acceptor	0.000	-4.213	0.000	0.000
EC 1.7.3 Oxidoreductases acting on other nitrogenous compounds as donors with oxygen as acceptor	0.000	-1.760	0.000	0.000
EC 2.1.1 Transferases transferring one-carbon groups methyltransferases	0.000	-2.477	0.000	0.000
EC 2.2.1 Transferases transferring aldehyde or ketonic groups transketolases and transaldolases	0.000	-3.703	0.000	0.000
EC 2.3.1 Transferases acyltransferases transferring groups other than amino-acyl groups	0.000	0.000	0.000	0.000
EC 2.3.3 Transferases acyltransferases acyl groups converted into alkyl groups on transfer	0.000	-7.409	0.000	0.000

Table S6.1 – continued from previous page

feature	LASSO	RIDGE	LARS	LASSO-LARS
EC 2.4.1 Transferases glycosyltransferases hexosyltransferases	0.000	-1.508	0.000	0.000
EC 2.4.2 Transferases glycosyltransferases pentosyltransferases	0.000	-1.161	0.000	0.000
EC 2.6.1 Transferases transferring nitrogenous groups transaminases	0.000	-3.791	0.000	0.000
EC 2.7.1 Transferases transferring phosphorus-containing groups phosphotransferases with an alcohol group as acceptor	0.000	-1.107	0.000	0.000
EC 2.7.2 Transferases transferring phosphorus-containing groups phosphotransferases with a carboxy group as acceptor	3.732	2.104	0.644	1.701
EC 2.7.3 Transferases transferring phosphorus-containing groups phosphotransferases with a nitrogenous group as acceptor	0.000	-6.624	0.000	0.000
EC 3.1.1 Hydrolases acting on ester bonds carboxylic ester hydrolases	1.012	-0.534	0.000	0.000
EC 3.1.2 Hydrolases acting on ester bonds thiolester hydrolases	0.000	-4.299	0.000	0.000
EC 3.1.3 Hydrolases acting on ester bonds phosphoric monoester hydrolases	-0.040	-4.226	0.000	0.000
EC 3.11.1 Hydrolases acting on carbon-phosphorus bonds acting on carbon-phosphorus bonds	0.000	-6.281	0.000	0.000
EC 3.2.1 Hydrolases glycosylases glycosidases, i.e. enzymes hydrolyzing o- and s-glycosyl compounds	0.000	-3.429	0.000	0.000
EC 3.4.11 Hydrolases acting on peptide bonds (peptidases) aminopeptidases	0.000	-1.535	0.000	0.000

Table S6.1 – continued from previous page

feature	LASSO	RIDGE	LARS	LASSO-LARS
EC 3.4.13 Hydrolases acting on peptide bonds (peptidases) dipeptidases	0.000	-6.667	0.000	0.000
EC 3.4.21 Hydrolases acting on peptide bonds (peptidases) serine endopeptidases	0.000	-2.286	0.000	0.000
EC 3.5.1 Hydrolases acting on carbon-nitrogen bonds, other than peptide bonds in linear amides	0.000	-5.764	0.000	0.000
EC 3.5.99 Hydrolases acting on carbon-nitrogen bonds, other than peptide bonds in other compounds	0.000	-0.695	0.000	0.000
EC 3.6.1 Hydrolases acting on acid anhydrides in phosphorus-containing anhydrides	0.000	2.272	0.000	0.000
EC 3.8.1 Hydrolases acting on halide bonds in c-halide compounds	0.000	-2.730	0.000	0.000
EC 4.1.1 Lyases carbon-carbon lyases carboxy-lyases	0.000	-5.351	0.000	0.000
EC 4.1.2 Lyases carbon-carbon lyases aldehyde-lyases	0.000	-1.126	0.000	0.000
EC 4.1.3 Lyases carbon-carbon lyases oxo-acid-lyases	0.382	-1.885	0.000	0.051
EC 4.2.1 Lyases carbon-oxygen lyases hydro-lyases	-0.008	-2.076	0.000	0.000
EC 4.2.3 Lyases carbon-oxygen lyases acting on phosphates	0.000	-4.873	0.000	0.000
EC 4.3.1 Lyases carbon-nitrogen lyases ammonia-lyases	-3.297	-8.734	-0.840	-1.553
EC 4.3.2 Lyases carbon-nitrogen lyases lyases acting on amides, amidines, etc	-0.226	-6.572	0.000	0.000

Table S6.1 – continued from previous page

feature	LASSO	RIDGE	LARS	LASSO-LARS
EC 4.4.1 Lyases carbon-sulfur lyases carbon-sulfur lyases	0.000	-7.543	0.000	0.000
EC 5.1.1 Isomerases racemases and epimerases acting on amino acids and derivatives	0.000	-5.305	0.000	0.000
EC 5.1.3 Isomerases racemases and epimerases acting on carbohydrates and derivatives	0.000	-0.551	0.000	0.000
EC 5.3.1 Isomerases intramolecular oxidoreductases interconverting aldoses and ketoses	0.277	-4.889	0.000	0.000
EC 5.4.2 Isomerases intramolecular transferases phosphotransferases (phosphomutases)	0.000	-5.016	0.000	0.000
EC 5.4.4 Isomerases intramolecular transferases transferring hydroxy groups	0.000	-4.959	0.000	0.000
EC 6.3.4 Ligases forming carbon-nitrogen bonds other carbon–nitrogen ligases	-4.079	-1.725	0.000	-2.735
EC 6.4.1 Ligases forming carbon-carbon bonds forming carbon-carbon bonds	0.000	-1.725	0.000	0.000
ALA%	0.000	-4.542	0.000	0.000
CYS%	0.505	0.637	0.000	0.000
ASP%	-1.020	-3.614	0.000	0.000
GLU%	0.000	5.881	0.000	0.000
PHE%	0.000	-1.869	0.000	0.000
GLY%	1.176	0.531	0.000	0.000
HIS%	0.000	-2.782	0.000	0.000
ILE%	0.510	1.589	0.000	0.000
LYS%	0.000	-2.409	0.000	0.000
LEU%	0.000	3.445	0.000	0.000
MET%	-1.580	-3.093	0.000	0.000

Table S6.1 – continued from previous page

feature	LASSO	RIDGE	LARS	LASSO-LARS
ASN%	0.000	1.677	0.000	0.000
PRO%	0.000	0.982	0.000	0.000
GLN%	0.000	-1.704	0.000	0.000
ARG%	0.000	0.322	0.000	0.000
SER%	0.000	-1.231	0.000	0.000
THR%	0.000	2.595	0.000	0.000
VAL%	0.000	2.756	0.000	0.000
TRP%	0.645	5.808	0.000	0.000
TYR%	-1.399	-3.940	-0.440	-1.440
Number of acidic amino-acids	0.000	2.302	0.000	0.000
aromaticity	0.000	-1.515	0.000	0.000
Backbone Clash Delta G	0.000	-0.600	0.000	0.000
Backbone H-bound Delta G	0.000	-3.744	0.000	0.000
Number of basic amino-acids	-1.029	-3.645	0.000	0.000
charge	0.000	0.000	0.000	0.000
Cis Peptide Bond Delta G	0.000	5.393	0.000	0.321
AUC for coils disorder	0.000	2.998	0.000	0.000
Disulfide bonds Delta G	0.000	2.109	0.000	-0.060
Electrostatics Delta G	0.000	-3.110	0.000	0.000
Ionisation Energy Contribution	0.190	-0.785	0.000	0.000
Hydrophobic groups Contribution	0.000	3.066	0.000	0.000
Burying polar groups penalty	0.000	-4.030	0.000	0.000
Torsion Delta G	-3.127	-2.143	-0.994	-2.030
VanderWaals Delta G	0.000	4.216	0.000	0.000
Energy penalization due to VanderWaals' clashes	0.000	-1.501	0.000	0.000
Main chain Delta H	0.000	3.373	0.000	0.000
Side Chain Delta H	0.000	-4.776	0.000	0.000

Table S6.1 – continued from previous page

feature	LASSO	RIDGE	LARS	LASSO-LARS
Grand Average of Hydropathy	0.000	1.334	0.000	0.000
Helix Dipole electrostatic contribuition	0.000	-3.308	0.000	0.000
Hot Loops	0.000	2.268	0.000	0.000
Instability Index	-1.011	-2.544	-0.255	-1.019
Electrostatic constant	0.000	-1.362	0.000	0.000
Mol Weight	0.000	-4.759	0.000	0.000
Non Polar	0.000	0.749	0.000	0.000
Number Of Products	-0.106	-3.103	0.000	0.000
Number Of Substrates	0.000	-3.935	0.000	0.000
Number of polar amino-acids	0.000	-0.754	0.000	0.000
AUC for rem465 disorder	0.000	-4.267	0.000	0.000
Structural Active Site Charge	0.000	3.702	0.000	0.000
Sidechain H-bound Delta G	0.000	0.246	0.000	0.000
Total Delta G	0.000	2.318	0.000	0.000

Table S6.2: Feature weight for enzymatic features using different machine-learning models. We used for regression methods: LASSO, ridge, LARS, LASSO-LARS

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Change in generic amino acid (not glycine)	0.000	-7.607	0.000	0.000
Change in aldehyde	-1.393	-1.938	-3.895	-3.681
Change in alkyl carbon	0.000	3.227	0.000	0.415
Change in amide	0.000	3.131	0.000	0.000
Change in anhydrides (except formic anhydride)	0.000	-0.846	0.000	0.000
Change in Aromatic Bond	0.000	-0.408	0.000	0.000
Change in azole	0.000	1.915	0.000	0.000
Change in carbo-thioester	0.000	2.083	0.000	0.956
Change in carbamate	0.000	0.682	0.000	0.000
Change in carbamic acid	0.000	0.960	0.000	0.000
Change in carbamic ester	0.000	-1.237	0.000	0.000
Change in Carbon	0.000	1.918	0.000	0.000
Change in carbonic acid/acid-ester	0.000	1.666	8.304	5.893
Change in carbonic acid/ester	0.000	1.666	0.000	0.000
Change in carbonyl group	0.000	-1.599	0.000	0.000
Change in carboxylate ion	0.000	-2.096	0.000	0.000
Change in carboxylic acid/conjugate_base	0.000	3.506	0.000	0.000
Change in carobnyl with carbon	0.000	1.710	0.000	0.000
Change in carobnyl with nitrogen	0.000	-10.618	0.000	0.000
Change in carobnyl with oxigen	0.000	3.300	0.000	0.000
Change in charge	0.000	10.777	0.000	0.000
Change in Charged Atoms	0.000	-7.145	0.000	0.000
Change in dicarboximide	0.000	0.682	0.000	0.000
Change in Double Bond	0.000	1.251	0.000	0.000
Change in enanmine	0.000	1.597	0.000	0.000
Change in ether	0.000	-3.278	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Change in glycine	0.000	-6.462	0.000	0.000
Change in hydrogen-bond acceptor	-2.060	-2.701	0.000	0.000
Change in hydrogen-bond donor	0.000	-5.695	0.000	0.000
Change in halide	0.000	-0.972	0.000	0.000
Change in halogen	0.000	-2.337	0.000	0.000
Change in Hydrogen Aceptors	0.000	-0.320	0.000	0.000
Change in Hydrogen Donors	0.000	3.851	0.000	0.000
Change in hydroxyl	0.000	1.341	0.000	0.000
Change in imine	0.000	0.055	0.000	0.000
Change in ketone	0.000	-2.778	0.000	0.000
Change in n-oxide	0.000	-2.357	0.000	0.000
Change in Negatively Charged Atoms	0.000	1.985	0.000	0.000
Change in nitrile	0.000	-1.296	0.000	0.000
Change in nitro	0.000	-1.760	0.000	0.000
Change in Nitrogen	0.000	2.060	0.000	0.000
Change in nitroso	0.000	1.760	0.000	0.000
Change in not monohydrogenated	0.000	3.455	0.000	0.000
Change in non-polar surface area	0.000	0.610	0.000	0.000
Change in Oxygen	0.000	-5.647	0.000	0.000
Change in peroxyde	0.000	-2.091	0.000	0.000
Change in Phosphate	0.000	2.344	0.000	0.000
Change in phosphoric acid	0.000	-1.756	0.000	0.000
Change in phosphoric ester	0.000	2.667	0.000	0.000
Change in Polarizability	0.000	5.706	2.509	4.666
Change in Positively Charged Atoms	-0.568	-12.120	0.000	0.000
Change in primary or secondary amine	0.000	-1.895	0.000	0.000
Change in primary_amine	0.000	-2.963	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Change in polar surface area	0.000	1.593	0.000	0.000
Change in Rotatable Bonds	0.000	-0.726	0.000	0.000
Change in Single Bond	0.000	2.033	0.000	0.000
Change in sp2 aromatic carbon	0.000	-1.289	0.000	0.000
Change in sp3 nitrogen	0.000	0.969	0.000	0.000
Change in sulfide	0.000	2.083	0.000	0.000
Change in thiol	0.000	-2.083	0.000	0.000
Change in Triple Bond	0.000	-1.296	0.000	0.000
Change in vinylic carbon	0.000	2.002	0.000	0.000
Delta Metabolites	0.909	-0.831	0.346	0.027
Products number of generic amino acid (not glycine)	0.000	6.761	0.000	0.000
Products number of acyl halide	0.000	0.510	0.000	0.000
Products number of aldehyde	1.286	1.580	0.355	0.882
Products number of alkyl carbon	0.000	-4.302	0.000	0.000
Products number of amide	0.000	1.768	0.000	0.000
Products number of anhydrides (except formic anhydride)	1.107	2.336	0.000	1.698
Products number Aromatic Bond	0.000	-0.717	0.000	0.000
Products number of azole	0.000	-1.960	0.000	0.000
Products number of carbo-thioester	0.000	3.084	0.000	0.000
Products number of carbamate	0.000	-1.237	0.000	0.000
Products number of carbamic acid	0.000	-1.237	0.000	0.000
Products number Carbon	0.000	-1.372	0.000	0.000
Products number of carbonic acid/acid-ester	0.000	-0.488	0.000	0.000
Products number of carbonic acid/ester	0.000	-0.488	0.000	0.000
Products number of carbonyl group	0.000	1.028	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Products number of carboxylate ion	-1.563	-0.526	0.000	-0.676
Products number of carboxylic acid/conjugate_base	0.000	-1.042	0.000	0.000
Products number of carobnyl with carbon	0.009	-4.427	0.000	0.000
Products number of carobnyl with nitrogen	0.000	6.692	0.000	0.000
Products number of carobnyl with oxigen	0.000	-0.458	0.000	0.000
Products charge	0.000	-1.758	0.000	0.000
Products number Charged Atoms	0.000	0.483	0.000	0.000
Products number Double Bond	0.000	1.780	0.000	0.000
Products number of enanmine	0.000	-3.394	0.000	0.000
Products number of ether	0.000	1.500	0.000	0.000
Products number of glycine	0.000	-1.081	0.000	0.000
Products number of hydrogen-bond acceptor	2.296	0.358	0.000	0.629
Products number of hydrogen-bond donor	0.000	0.996	0.000	0.000
Products number of halide	0.000	-2.541	0.000	0.000
Products number of halogen	0.000	-1.016	0.000	0.000
Products number Hydrogen Aceptors	0.000	3.062	0.000	0.000
Products number Hydrogen Donors	0.000	-1.079	0.000	0.000
Products number of hydroxyl	0.000	3.316	0.000	0.000
Products number of imine	0.000	1.348	0.000	0.000
Products number of ketone	0.973	6.403	0.000	0.000
Products number Negatively Charged Atoms	0.000	-0.076	0.000	0.000
Products number of nitrile	0.000	1.180	0.000	0.000
Products number of nitro	0.000	2.918	0.000	0.000
Products number Nitrogen	0.000	2.244	0.000	0.000
Products number of nitroso	0.000	-1.760	0.000	0.000
Products number of not monohydrogenated	0.000	-0.702	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Products number Npsa Mean	0.000	0.997	0.000	0.000
Products number Oxygen	0.000	2.805	0.000	0.000
Products number of peroxyde	0.000	4.001	0.000	0.000
Products number Phosphate	0.000	0.578	0.000	0.000
Products number of phosphoric acid	-1.633	2.729	0.000	0.000
Products number of phosphoric ester	-0.721	-3.484	0.000	0.000
Products number Polarizability Mean	0.000	-12.162	0.000	0.000
Products number Positively Charged Atoms	0.000	1.326	0.000	0.000
Products number of primary or secondary amine	0.000	-4.840	0.000	0.000
Products number of primary_amine	0.000	5.677	0.000	0.000
Products number Psa Mean	0.000	-5.482	0.000	0.000
Products number Rotatable Bonds	0.000	-1.184	0.000	0.000
Products number Single Bond	0.000	-1.228	0.000	0.000
Products number of sp2 aromatic carbon	0.000	0.638	0.000	0.000
Products number of sp3 nitrogen	0.000	2.570	0.000	0.000
Products number of sulfide	0.000	2.936	0.000	0.000
Products number Sulfur	0.000	2.666	0.000	0.000
Products number of thiol	0.000	-1.081	0.000	0.000
Products number Triple Bond	0.000	1.180	0.000	0.000
Products number of vinylic carbon	0.000	-1.712	0.000	0.000
Substrates number of generic amino acid (not glycine)	0.000	4.563	0.000	0.000
Substrates number of acyl halide	0.000	0.510	0.000	0.000
Substrates number of aldehyde	-0.040	-2.117	0.000	0.000
Substrates number of alkyl carbon	0.000	-3.461	0.000	0.000
Substrates number of amide	0.000	2.355	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Substrates number of anhydrides (except formic anhydride)	2.096	0.644	0.000	1.537
Substrates number of Aromatic Bond	0.000	-0.953	0.000	0.000
Substrates number of azole	0.000	-0.763	0.000	0.000
Substrates number of carbo-thioester	0.000	3.625	0.000	0.000
Substrates number of carbamate	0.000	-0.555	0.000	0.000
Substrates number of carbamic acid	0.000	0.682	0.000	0.000
Substrates number of carbamic ester	0.000	-1.237	0.000	0.000
Substrates number of Carbon	0.000	-1.161	0.000	0.000
Substrates number of carbonic acid/acid-ester	6.044	2.845	0.000	4.421
Substrates number of carbonic acid/ester	0.000	2.845	0.000	0.000
Substrates number of carbonyl group	0.000	0.373	0.000	0.000
Substrates number of carboxylate ion	-3.198	-4.980	-2.055	-2.279
Substrates number of carboxylic acid/conjugate_base	0.000	2.464	0.000	0.000
Substrates number of carobnyl with carbon	0.000	-2.937	0.000	0.000
Substrates number of carobnyl with nitrogen	0.000	2.711	0.000	0.000
Substrates number of carobnyl with oxigen	0.000	2.842	0.000	0.000
Substrates charge	0.000	1.480	0.000	0.000
Substrates number of Charged Atoms	0.000	-0.467	0.000	0.000
Substrates number of dicarboximide	0.000	0.682	0.000	0.000
Substrates number of Double Bond	0.000	2.160	0.000	0.000
Substrates number of enanmine	0.000	-0.200	0.000	0.000
Substrates number of ether	0.000	-0.139	0.000	0.000
Substrates number of glycine	0.000	-3.235	0.000	0.000
Substrates number of hydrogen-bond acceptor	0.000	-0.468	0.000	0.000
Substrates number of hydrogen-bond donor	0.000	-0.597	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Substrates number of halide	0.000	-3.027	0.000	0.000
Substrates number of halogen	0.000	-1.600	0.000	0.000
Substrates number of Hydrogen Aceptors	0.000	3.021	0.000	0.000
Substrates number of Hydrogen Donors	0.000	-0.165	0.000	0.000
Substrates number of hydroxyl	0.000	4.038	0.000	0.000
Substrates number of imine	0.000	1.376	0.000	0.000
Substrates number of ketone	0.000	0.847	0.000	0.000
Substrates number of n-oxide	0.000	-2.357	0.000	0.000
Substrates number of Negatively Charged Atoms	0.000	0.407	0.000	0.000
Substrates number of nitrile	0.000	0.964	0.000	0.000
Substrates number of nitro	0.000	1.158	0.000	0.000
Substrates number of Nitrogen	0.000	2.506	0.000	0.000
Substrates number of not monohydrogenated	0.000	-0.173	0.000	0.000
Substrates number of Npsa Mean	0.000	1.323	0.000	0.000
Substrates number of Oxygen	0.000	1.350	0.000	0.000
Substrates number of peroxyde	0.000	1.728	0.000	0.000
Substrates number of Phosphate	0.000	1.164	0.000	0.000
Substrates number of phosphoric acid	0.000	2.217	0.000	0.000
Substrates number of phosphoric ester	0.000	-2.595	0.000	0.000
Substrates number of Polarizability Mean	0.000	-7.784	0.000	0.000
Substrates number of Positively Charged Atoms	0.000	-1.592	0.000	0.000
Substrates number of primary or secondary amine	0.000	-6.104	0.000	0.000
Substrates number of primary_amine	0.000	3.899	0.000	0.000
Substrates number of Psa Mean	0.000	-1.975	0.000	0.000
Substrates number of Rotatable Bonds	0.000	-1.434	0.000	0.000
Substrates number of Single Bond	0.000	-0.828	0.000	0.000
Substrates number of sp2 aromatic carbon	0.000	-0.007	0.000	0.000

Table S6.2 – continued from previous page

Feature	LASSO	RIDGE	LARS	LASSO-LARS
Substrates number of sp3 nitrogen	0.000	2.719	0.000	0.000
Substrates number of sulfide	0.000	3.978	0.000	0.000
Substrates number of Sulfur	0.000	2.666	0.000	0.000
Substrates number of thiol	-0.478	-5.247	0.000	0.000
Substrates number of Triple Bond	0.000	0.964	0.000	0.000
Substrates number of vinylic carbon	0.000	-0.585	0.000	0.000

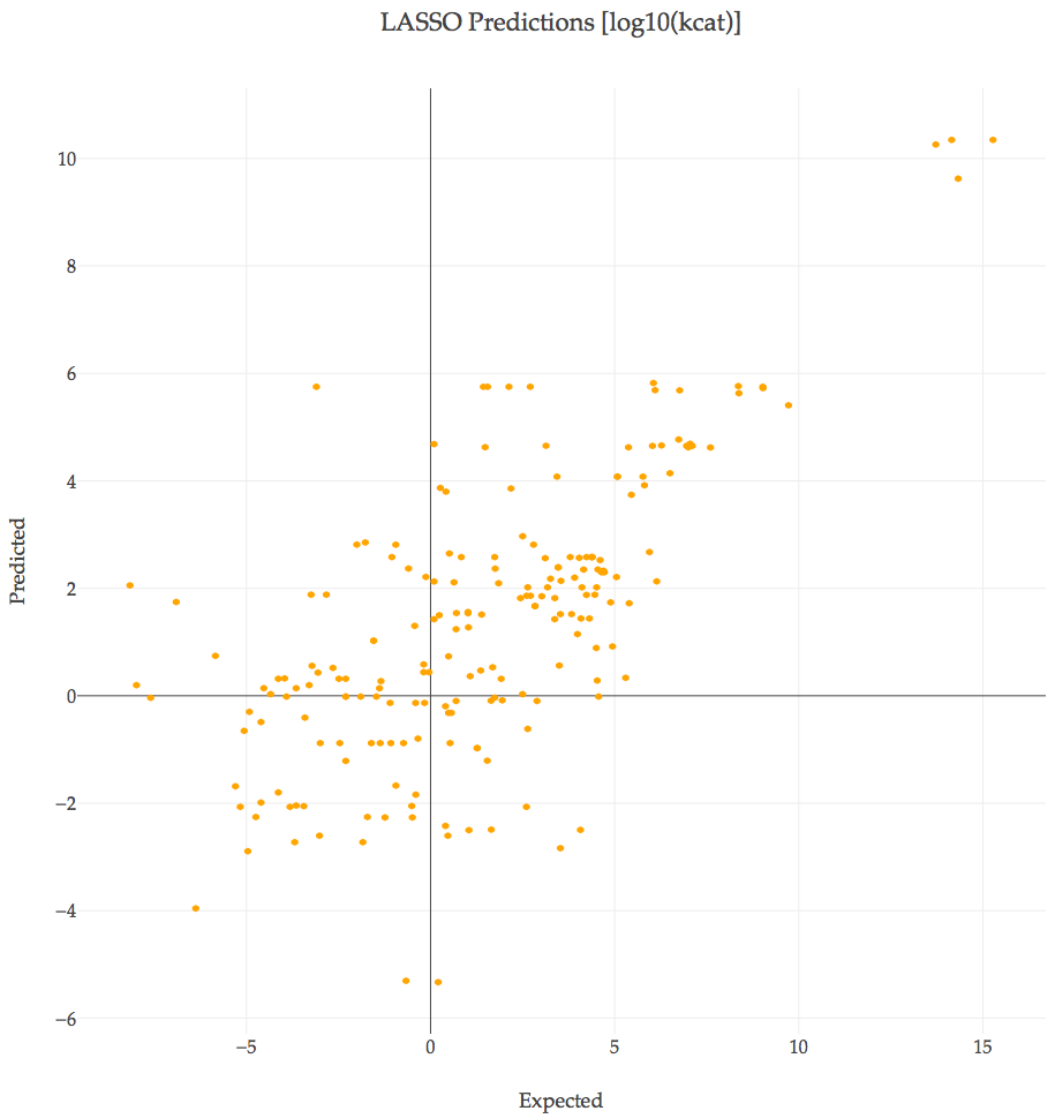


Figure S6.1: Linear model predictions fitted with LASSO. The x axis shows expected k_{cat} values and y axis shows the model predictions

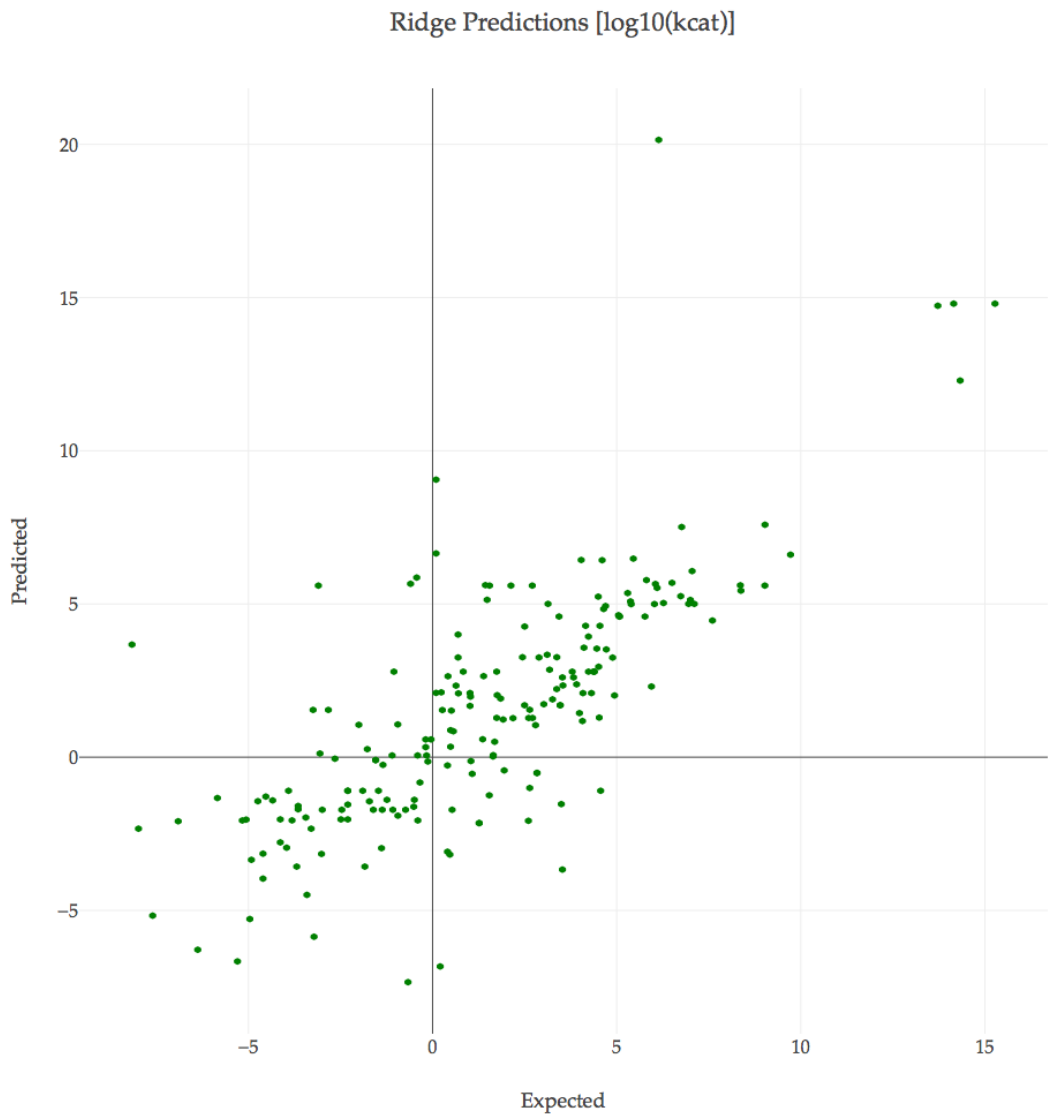


Figure S6.2: Linear model predictions fitted with ridge regression. The x axis shows expected k_{cat} values and y axis shows the model predictions

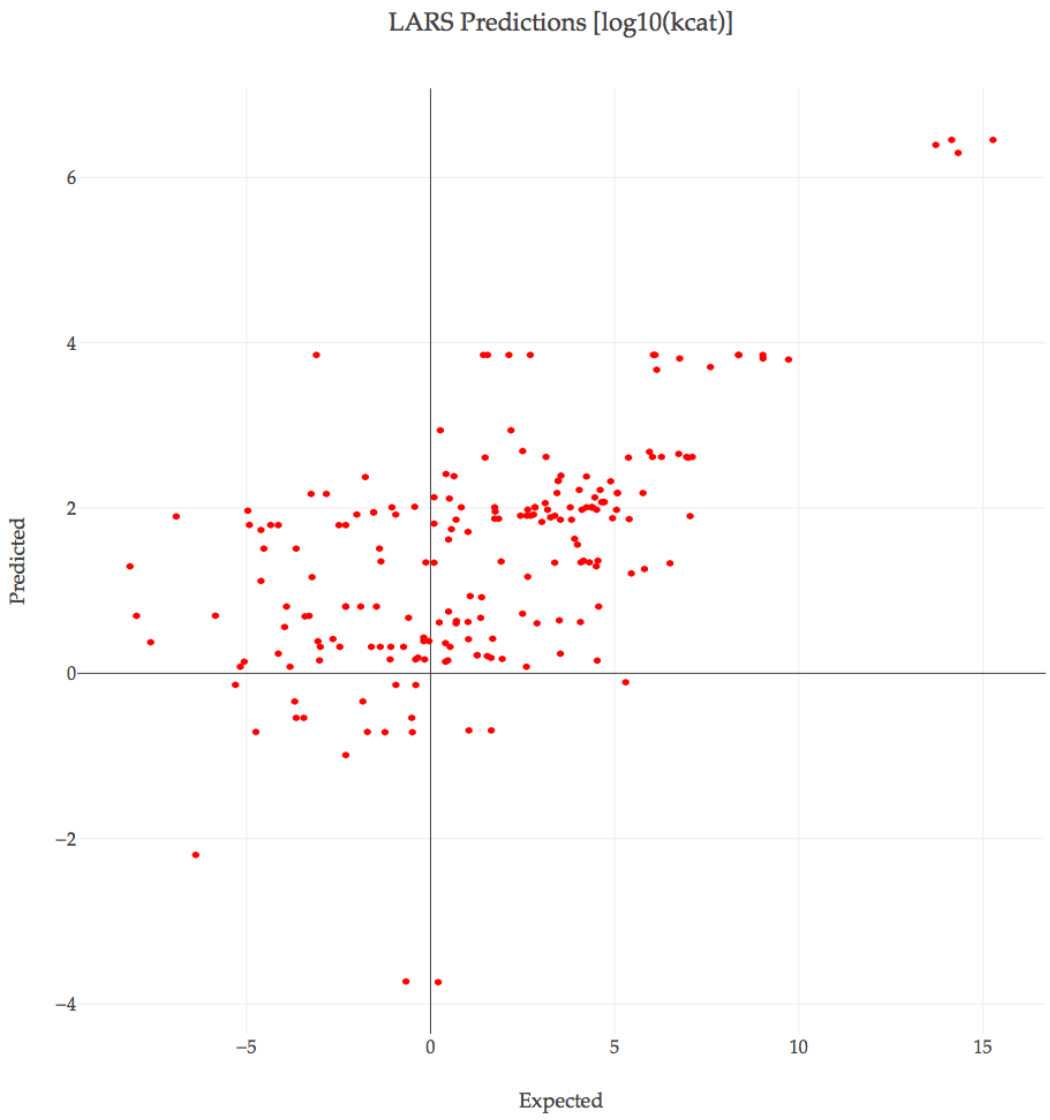


Figure S6.3: Linear model predictions fitted with LARS. The x axis shows expected k_{cat} values and y axis shows the model predictions

Figure S6.4: Linear model predictions fitted with LASSO-LARS. The x axis shows expected k_{cat} values and y axis shows the model predictions

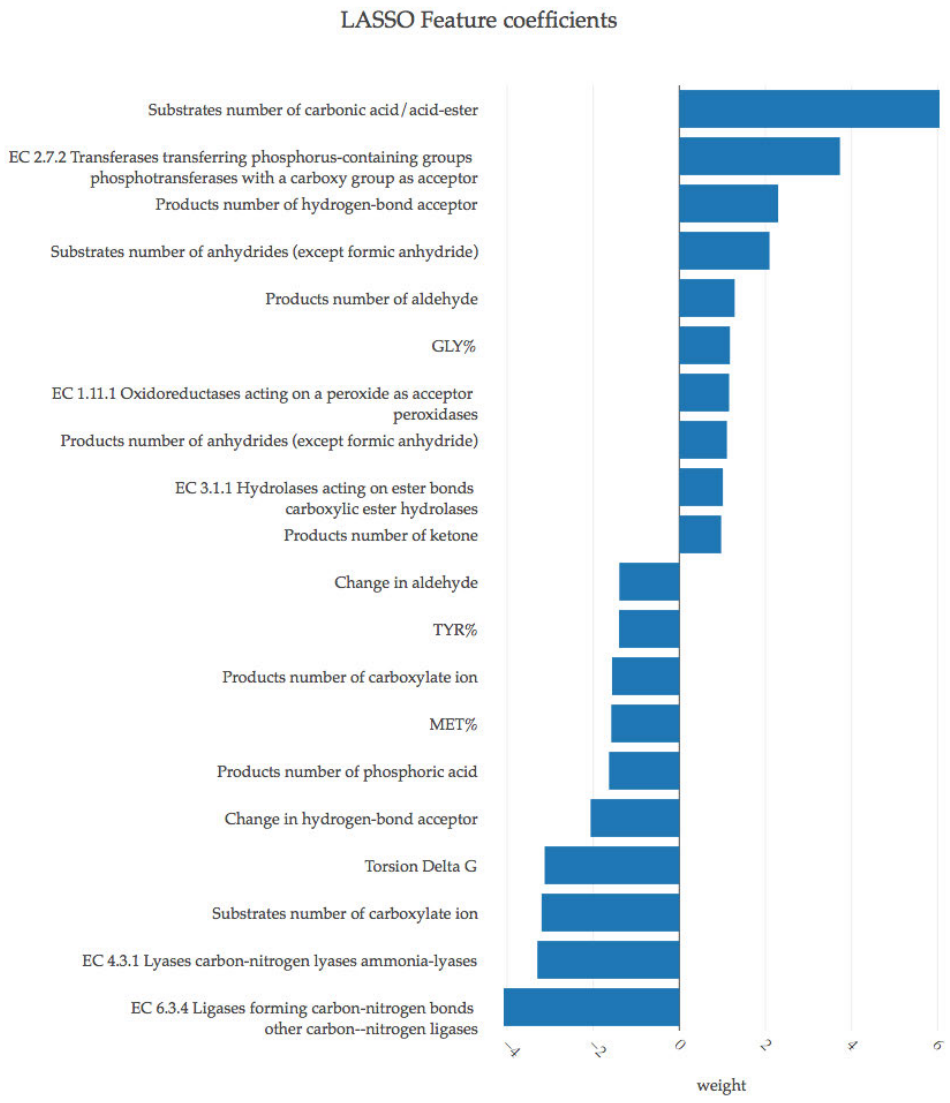


Figure S6.5: Top 20 coefficients resulting from LASSO.

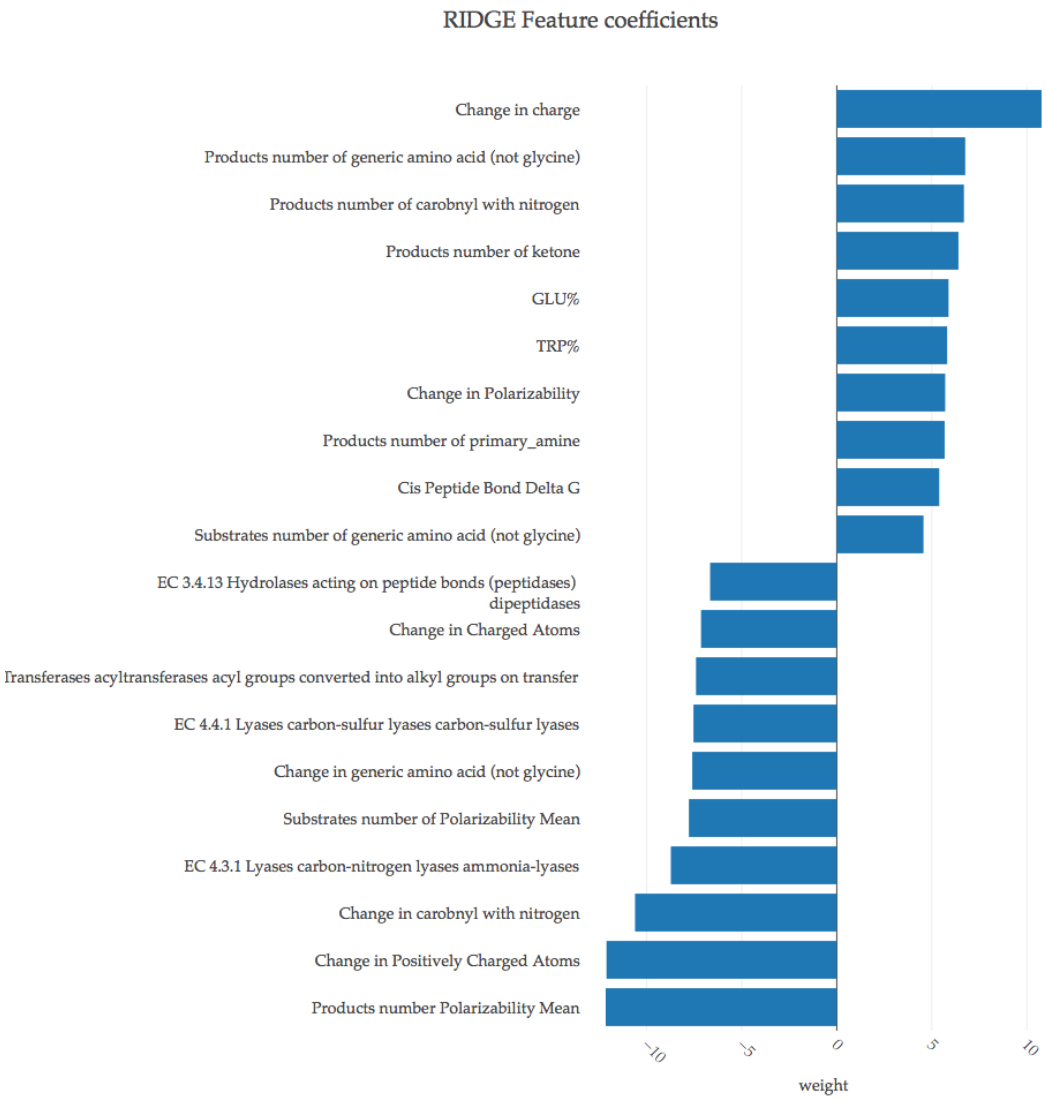


Figure S6.6: Top 20 coefficients resulting from ridge regression.

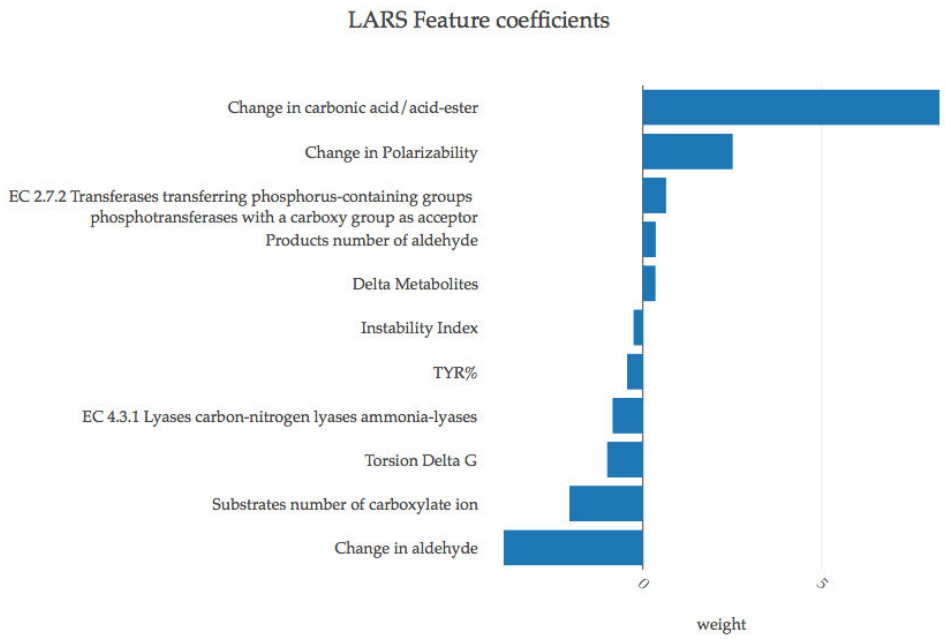


Figure S6.7: Top 20 coefficients resulting from LARS.

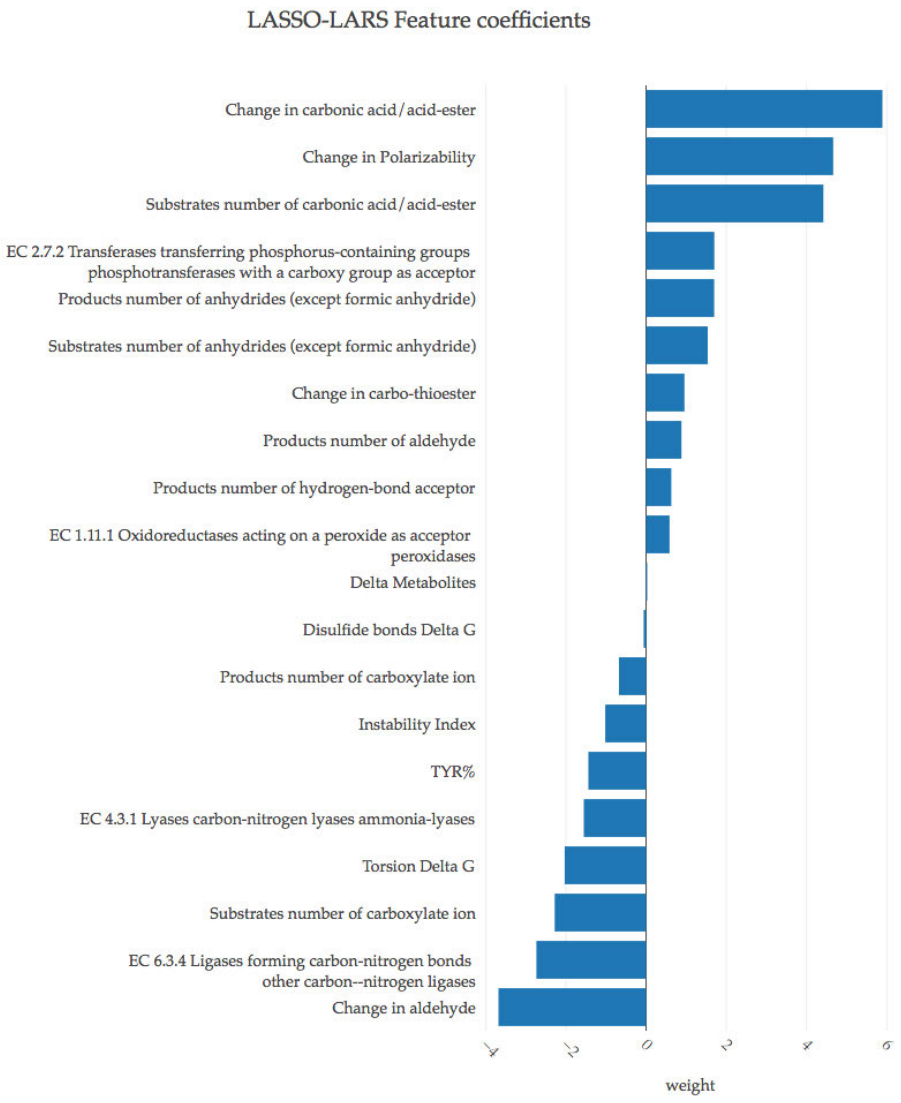


Figure S6.8: Top 20 coefficients resulting from LASSO-LARS.

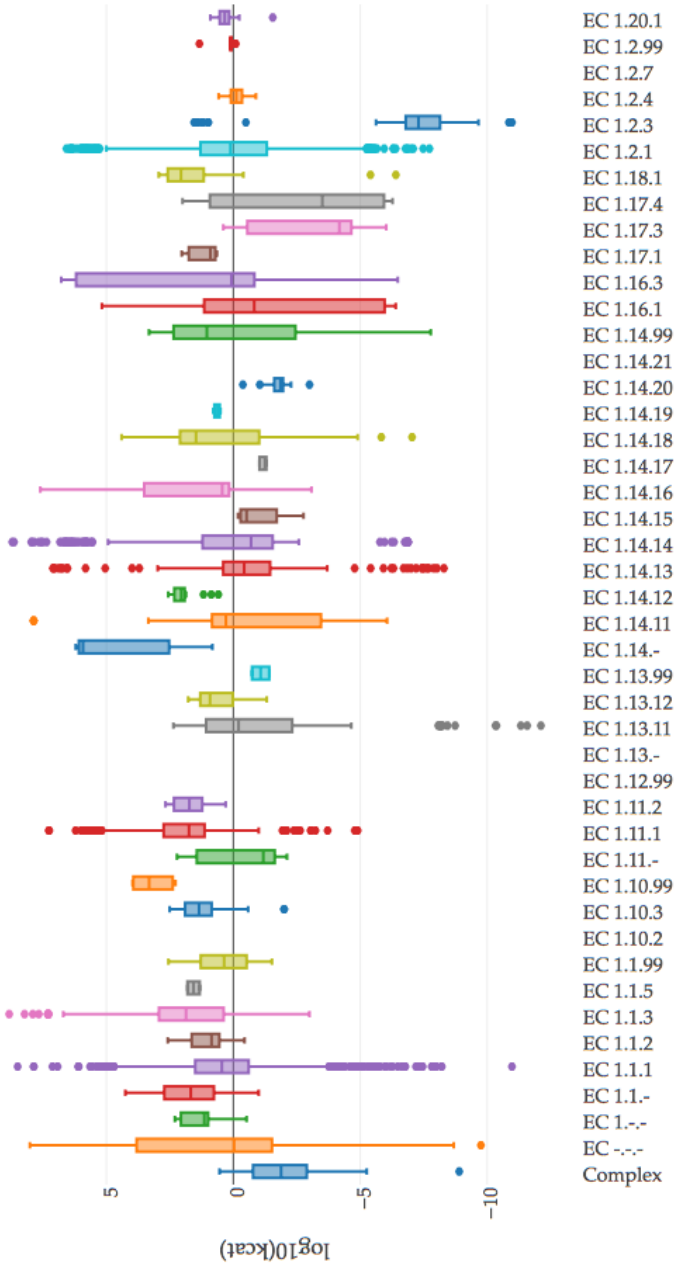


Figure S6.9: k_{cat} and EC numbers. This box plot shows the different EC numbers (up to the third digit) and the distribution of catalytic rate constants

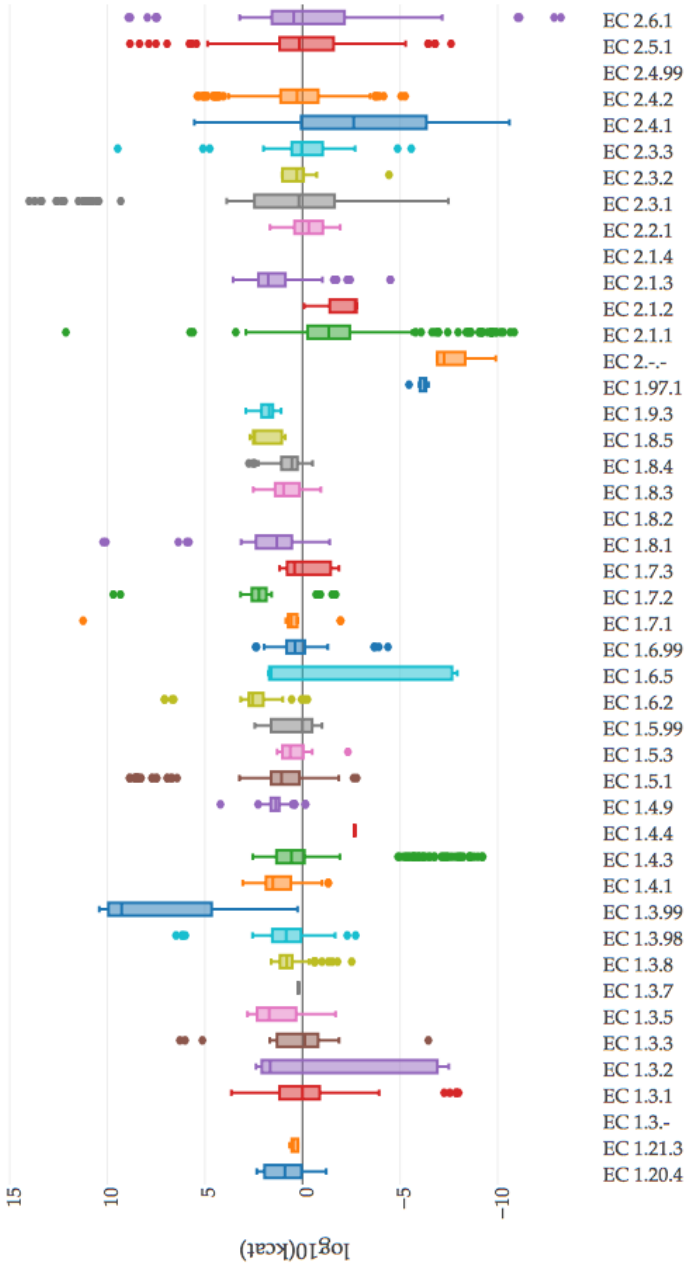


Figure S6.10: Continued. ***k*$_{cat}$ and EC numbers**. This box plot shows the different EC numbers (up to the third digit) and the distribution of catalytic rate constants

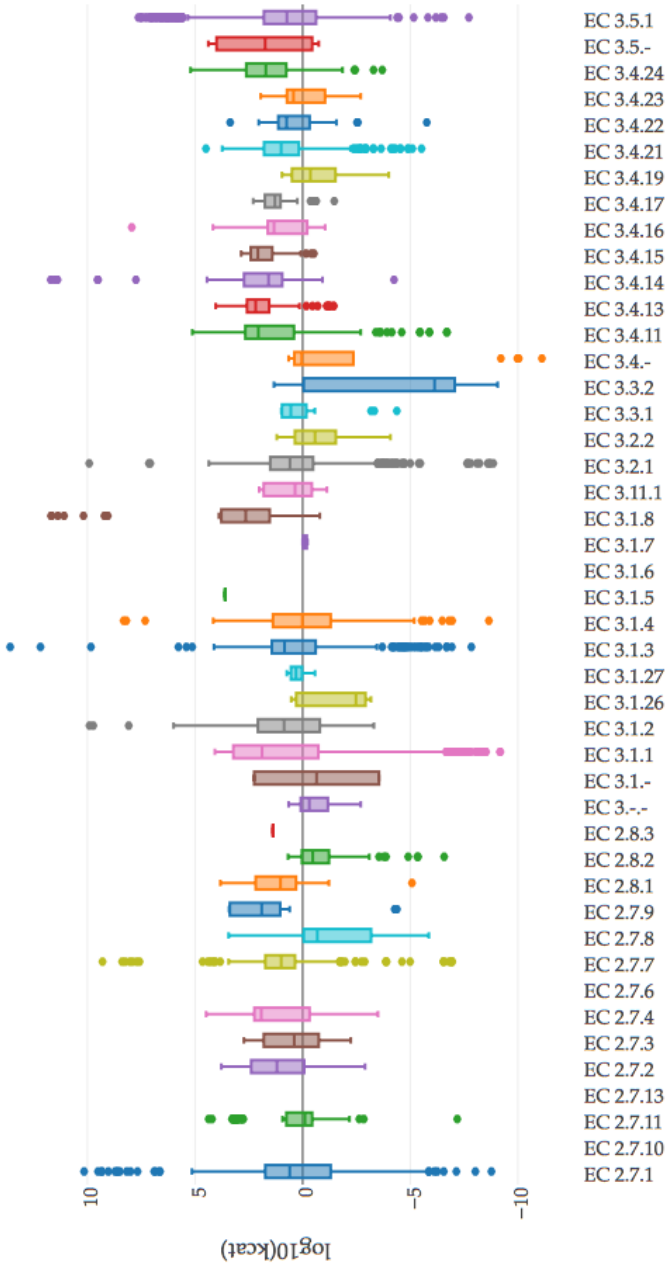


Figure S6.11: Continued. ***k_{cat}* and EC numbers.** This box plot shows the different EC numbers (up to the third digit) and the distribution of catalytic rate constants

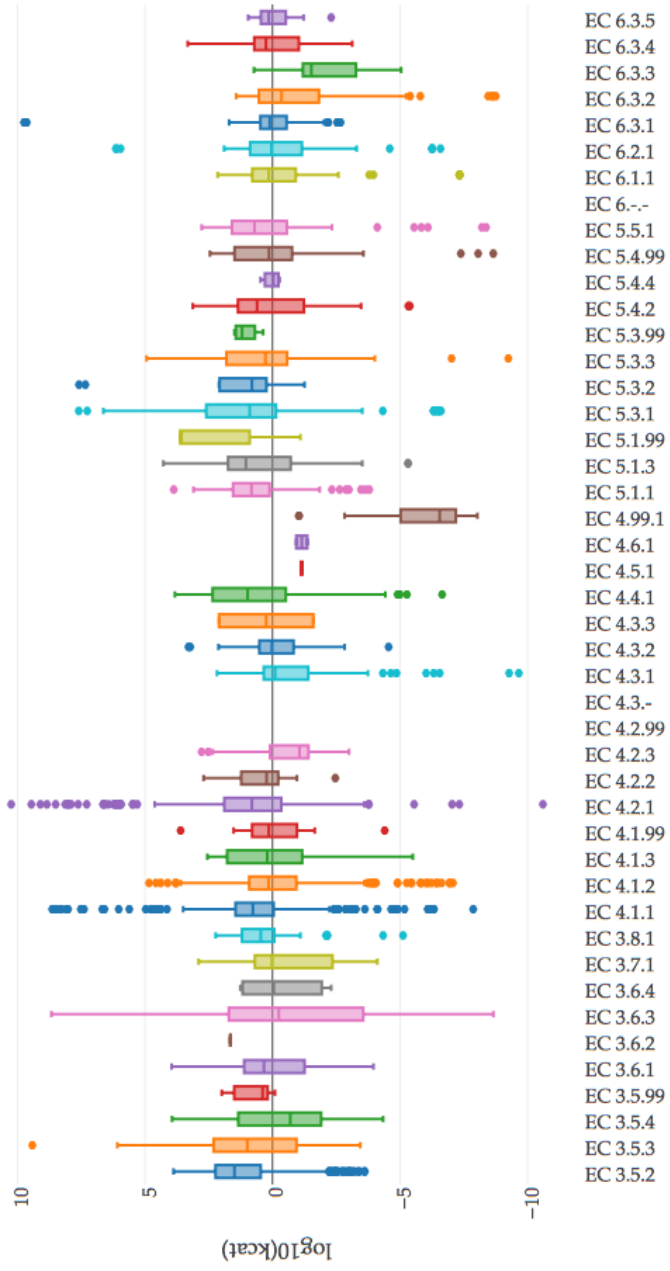


Figure S6.12: Continued. ***k*$_{cat}$ and EC numbers**. This box plot shows the different EC numbers (up to the third digit) and the distribution of catalytic rate constants

Figure S6.13: Pearson correlation coefficients (absolute values) between EC numbers and substrate features. The bars on the top show the number of data points for each EC number.

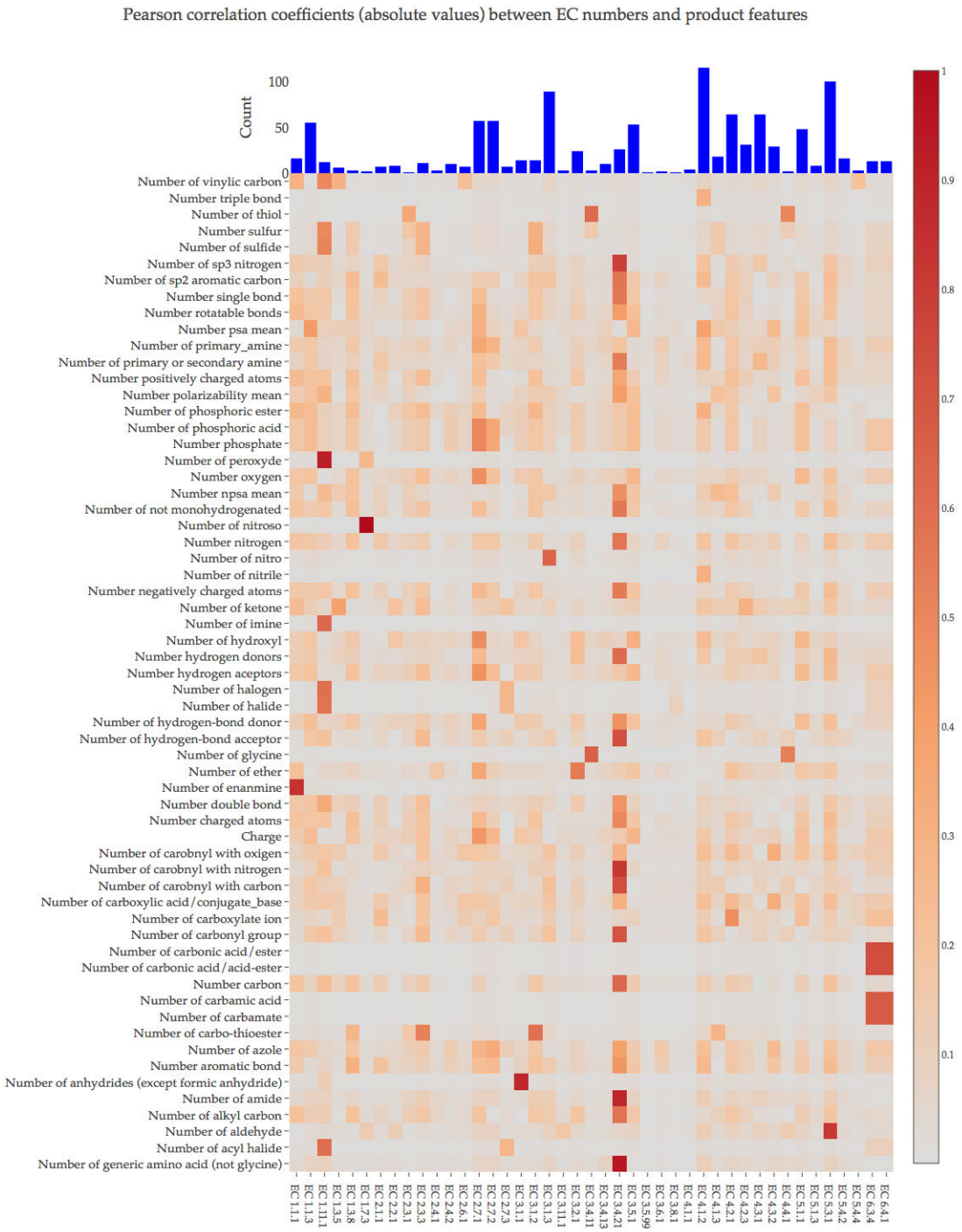


Figure S6.14: Pearson correlation coefficients (absolute values) between EC numbers and product features. The bars on the top show the number of data points for each EC number.

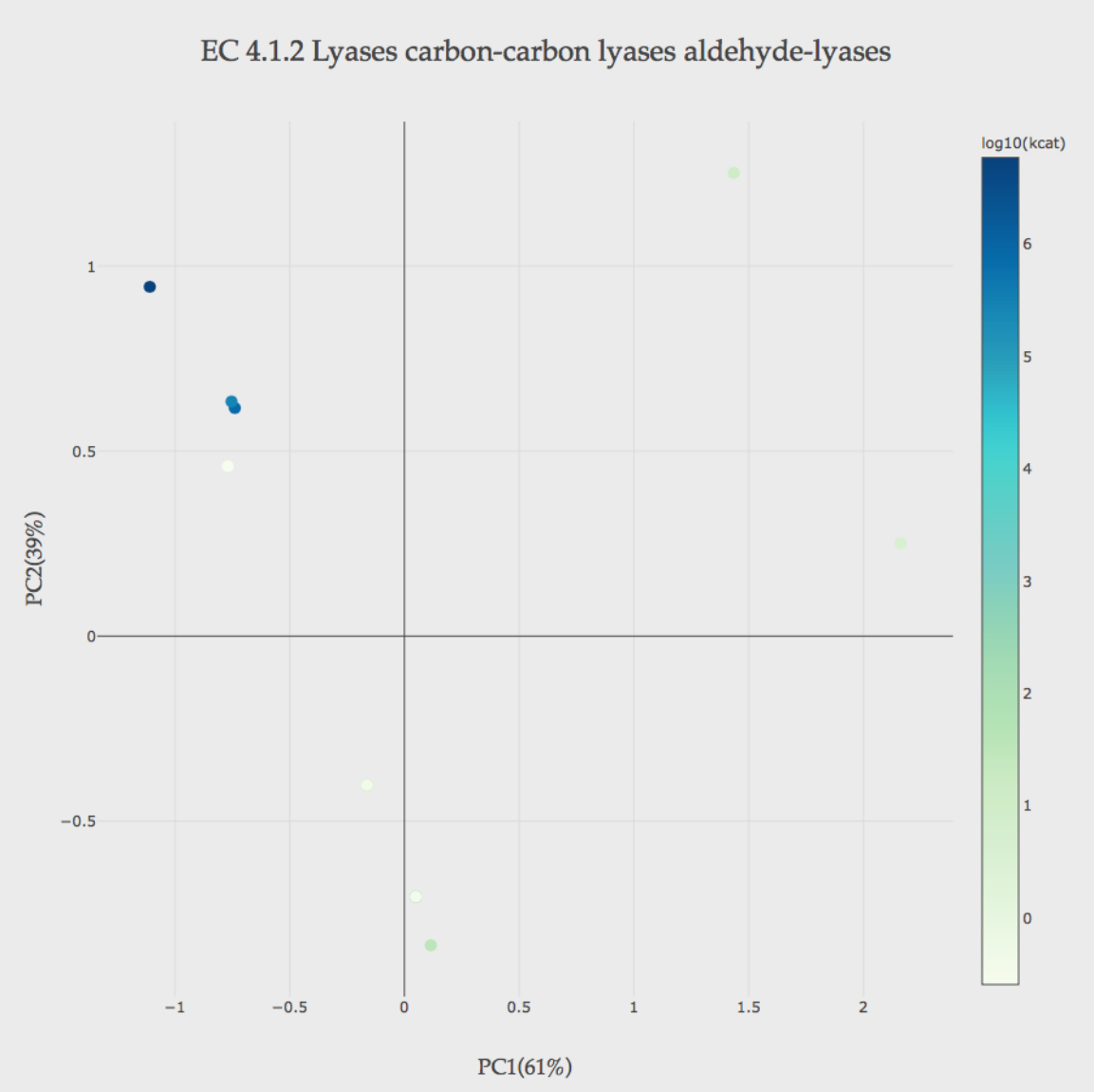


Figure S6.15: PCA decomposition of the data with EC number 4.1.2. On the left we show the coefficients of each feature in the first component.

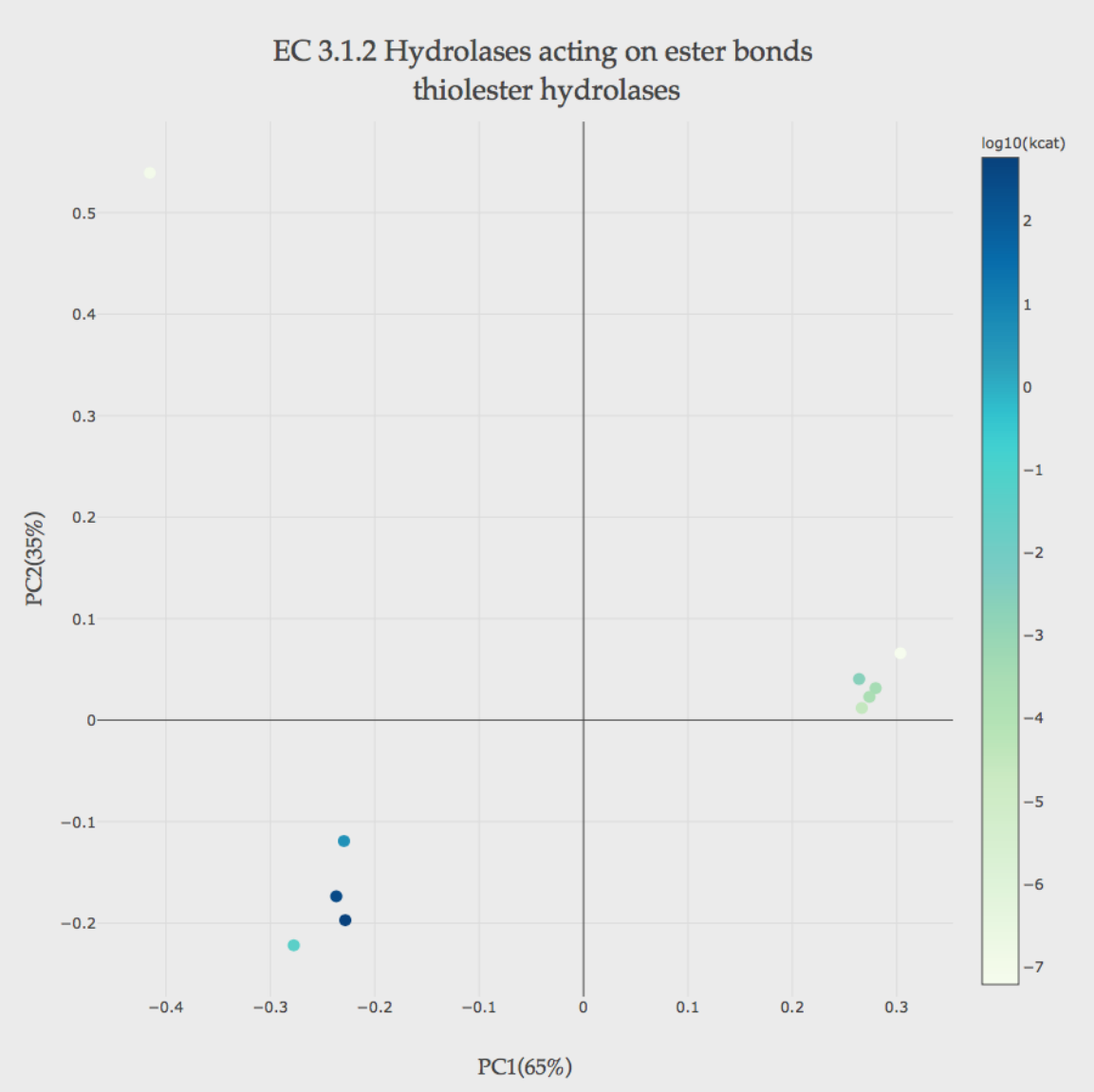


Figure S6.16: PCA decomposition of the data with EC number 3.1.2. On the left we show the coefficients of each feature in the first component.

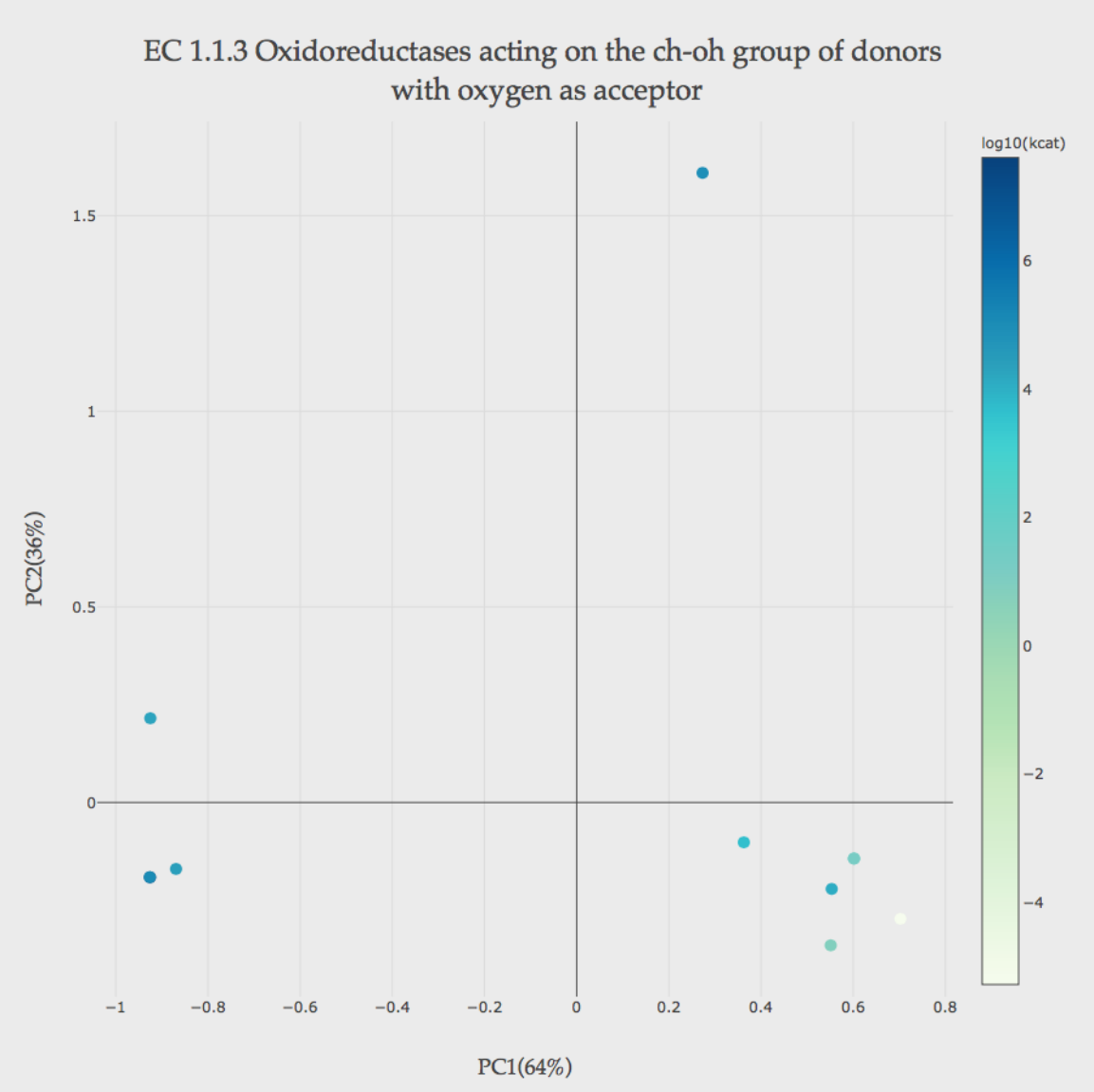


Figure S6.17: PCA decomposition of the data with EC number 1.1.3. On the left we show the coefficients of each feature in the first component.

A conclusion is the place where you got tired thinking.

Martin H. Fischer

7

Conclusions and perspectives

Computer-aided design using genome-scale metabolic models

Computer-aided design (CAD) software, like *cameo*, will enable more scientists to use state-of-the-art algorithms and advanced models of metabolism. *Cameo* overcomes limitations of the currently available software: it is easy to use and provides all the necessary algorithms for strain design. The software is truly open-source and free. We hosted the source code at GitHub (`https://github.com`), a platform that allows anyone to collaborate in the development of *cameo*. We also used a very permissive license, the Apache License 2.0 that allows anyone to use *cameo* for personal or commercial applications.

We implemented an easy-to-use predefined workflow in *cameo* to perform the strain design task for a target metabolite. The workflow will identify if the metabolite is native or not, identify heterologous pathways if necessary, and run different algorithms to find genetic engineering targets. The current workflow will search design strategies using *Escherichia coli* and *Saccharomyces cerevisiae* models, but one can provide their own GEM and still use the workflow. Users with minimal knowledge of constraint-based modeling can get create strain designs without the effort of running multiple different tools.

This software will continue to be developed in order to make *cameo* more complete and easier to use. We want to integrate *cameo* with genome-editing tools to convert *in silico* strain designs into laboratory protocols. We want to extend our database of heterologous reactions with new pathways identified using retrobiosynthesis. Finally, the designs generated with *cameo*, including the phenotypes, can be used to calculate strain performance (i.e., titers and productivity) when combined with models of the fermentation processes. A great feature of *cameo* is its modular design. It is built to accommodate new technologies. When new models and modeling approaches become available, we will be able to use them with *cameo*.

Cameo is already being use *in-house* to design strains and to teach students at the university. We have shown some capabilities of our software in this thesis. In collaboration with my colleagues, we designed a *S. cerevisiae* strain that can produce more mevalonate. Mevalonate is particularly interesting intermediate given the high number of valuable chemicals that can be produced using the mevalonate pathway. The designed strain features a shift in redox balance from

NADH to NADPH implemented in a previously . We used our strain design methods to identify key changes in the metabolic pathways and to test our hypothesis *in silico*.

In this thesis we also addressed a gap between academia and industry. With MARSI, our new algorithm, we now support design of evolutionary engineering strategies such as classical strain improvement (CSI) and adaptive laboratory evolution (ALE) that are widely used in industry. MARSI is an *in silico* strain design method that predicts metabolite targets, instead of gene or reaction knockouts. The metabolites can be targeted with chemical analogues, such as antimetabolites, to force changes in microbial phenotypes. This solution does not require rDNA technology to be implemented. This is a new step towards bringing constraint-based methods using GEMs to the food industry, that is highly restricted by genetically modified organism (GMO) legislation in Europe.

We were able to find metabolite targets that can replace gene and reaction knockouts in previously published strain designs. This shows that our method is capable of finding relevant engineering targets. Still, we lack experimental validation for new metabolite-based designs. The next step would testing the design strategies and metabolite analogues identified by MARSI using ALE. We have also already started the development of an advanced mathematical formulation to predict the effects of metabolite and analogue competing for enzyme active sites. This method will be able to simulate the effect of using chemical analogues for essential metabolites, such as amino-acids.

MARSI can also identify candidate analogues from a chemical database containing known drugs, analogues and toxic compounds. We successfully retrieved chemical analogues for a set of chemicals with known analogues. The chemicals identified by MARSI are more similar to the query chemicals than the known analogues. A limitation of our current database is the lack of toxicity, dosage and price information for the chemicals. Adding that information would be valuable for prioritization of possible analogues. First, pricing information can be used to exclude chemicals that are not for affordable or readily available in the market. Second, it could help identify the best chemical based on the toxicity and the experimental setup.

The next step to improve CAD is to create a good user interface. We described limitations of the available software and address those with *cameo*. Indeed, we provide an easy to use programmatic interface and a command line interface for deployment in HPC infrastructures. And while the methods are now available and within reach of many more users, work is still needed

to improve the interaction between users and software. We need to eliminate the low level interactions between users and mathematical programming, such as numerical precision, infeasibility, etc. The ultimate goal is to provide a graphical user interface (GUI), where users can navigate intuitively.

Implementing a strain design *in vivo*

We are in the process of building a *S. cerevisiae* strain with high mevalonate production based on the design. Preliminary results suggest that our *in silico* designs are leading us in the right direction. Still, building strains resulting from CAD methods is not a trivial task. Creating and testing multiple hypothesis in the computer can be quickly and easily done. But the experimental work part poses challenges: cloning heterologous enzymes and overcome native regulatory mechanisms.

While building our *S. cerevisiae* strain we encountered several bottlenecks. Finding the right level of expression is something a GEM can calculate from the reaction stoichiometry. We need to include protein abundance and enzyme kinetics to calculate the necessary levels of protein. We also need to identify the optimal combination of promoters, copy numbers, ribosome binding sites and terminators that yield the best expression and translation levels of our heterologous pathways.

Another limiting step is the host native regulation on the pathway we are trying to optimize. The mevalonate pathway is regulated by negative feedback, imposed by different products downstream of the pathway. Directed enzyme evolution or alternative heterologous enzymes can help us overcome the native regulation.

To finish this project we need to validate the performance of this strain against previously published and industrial strains. To do that, we are planning fed batch and continuous fermentation experiments. Also, further *in silico* analysis can be done to identify optimal growth and production rates and engineering strategies to increase the robustness of the strain (i.e., elimination of alternative pathways and reduce carbon).

Predicting kinetic parameters from sequence, structure and reaction features

Retrieving the k_{kcat} values for enzymes together with correct sequence and structural information proved to be a complex process. Despite the effort of creating standards for reporting enzymes, sequences, chemical compounds, and reactions, we observed a large amount of data without proper identifiers even in the most curated of databases. Another problem is the quality of the data reported such as the mutations registered in enzyme kinetics databases. We found a large number of entries with correct protein identifiers but the mutations reported did not match the peptide sequences.

Despite the problems identified in the data, we decided to explore the relationship between sequence and catalytic rates using the limited dataset that we had assembled. We identified no correlation between predicted effects of mutations based on sequence conservation and k_{kcat} value changes due to the mutations. However, when we used structural models, we found some cases where predicted changes in protein stability correlated with changes in catalytic rates. That means that we need to assess whether the protein concentration used in the experiments accounts for the unfolded/non-active enzymes. The integration of k_{kcat} and protein abundance into GEMs has been shown to improve *in silico* predictions of metabolic phenotypes at least in yeast. Still, consistently measuring the catalytic rates requires huge number of experiments, specially if we want to include genetic variation in the experimental design.

As an alternative use of the dataset we had collected, we built linear model to predict k_{cat}s using features from enzyme and the chemical reactions. We used 1012 samples, which represent only 2% of the kinetics data in SABIO-RK! (SABIO-RK!). While this amount of data is not enough to build a global predictive model, we observed that the chemical composition of certain substrates and products can be used to separate high and low k_{kat}s. This is an interesting result as it is not well understood what determines k_{kat}s for enzymes - the actual chemistry that is catalyzed by the enzyme or the cellular context of the enzyme (i.e. what rate is needed for the enzyme in the organism where it functions).

We identified reaction features that are unique for different EC numbers and that makes our feature matrix very sparse. We can now think about new approaches to analyze this data. We

can consider spliting our data into EC groups (i.e., oxireductases, transferases, hydrolases, lyases, isomerases and ligases) separately. Also, we can focus on addressing different parts of the problem differently. For example, use a model to analyze the effect of the temperature and pH in the enzyme stability. Then, using a different model, predict the interactions between enzymes and substrates. In the end, a group of combined models can be a practical solution to predict effects of different features including genetic variation on catalytic activities. Alternatively, we could collect sufficient data to apply deep-learning methods. But while deep-learning has shown to outperform simple models in different applications, it is very difficult to infer knowledge from them.

Despite the approaches that come next, we took a step forward in analyzing enzyme kinetics at a large scale and moved one step closer to connect the information encoded in the DNA sequences with our mathematical models. Still, there is a long road ahead to allow quantitative predictions of effects of sequence variation on metabolic phenotypes. For now, the focus of the scientific community should be on obtaining more and better data that relate sequence changes to functional changes in enzymes using approaches such as deep mutational scanning.

Colophon

This thesis was typeset using LaTeX, originally developed by Leslie Lamport and based on Donald Knuth's TeX. The body text is set in 11 point Arno Pro, designed by Robert Slimbach in the style of book types from the Aldine Press in Venice, and issued by Adobe in 2007. A template, which can be used to format a PhD thesis with this look and feel, has been released under the permissive MIT (XII) license, and can be found online at github.com/suchow/ or from the author at suchow@post.harvard.edu.